Cuando se trabaja con un ordenador, lo único que puede apreciarse, a simple vista, es una especie de caja negra que, misteriosamente, acepta una serie de instrucciones.

En realidad, un ordenador es una máquina capaz de recibir, transformar, almacenar y suministrar datos. En este libro se construye un pequeño ordenador, capaz de realizar dichas funciones, a partir de un microprocesador. Podrá observar de qué partes se compone el ordenador y cómo conectarlas entre sí. Además, podrá adentrarse en el mundo del «hardware» utilizándolo, por ejemplo, como un temporizador para controlar cualquier electrodoméstico casero.

Por otra parte, se ha pensado en la construcción de un ordenador dinámico. A partir del diseño básico, podrá ampliarlo con el único límite que impone la propia imaginación.

15 Cómo construir su propio ordenador

SC

ENCICLOPEDIA PRACTICA DE LA

INFORMATICA APLICADA

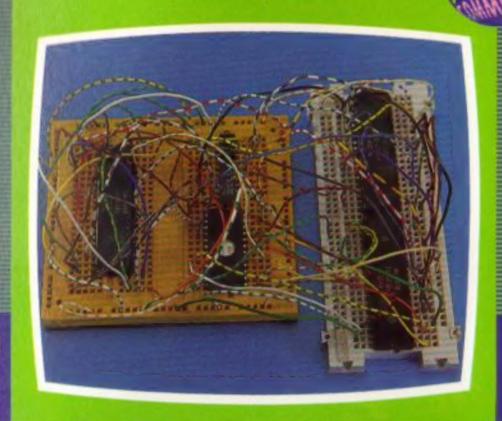
15

Cómo construir su propio ordenador

AIA



SINCLAIR AMSTRAD COMMODORE IBM



EDICIONES SIGLO CULTURAL

Una publicación de

EDICIONES SIGLO CULTURAL, S.A.

Director-editor:
RICARDO ESPAÑOL CRESPO.

Gerente:
ANTONIO G. CUERPO.

Directora de producción:
MARIA LUISA SUAREZ PEREZ.

Directores de la colección:
MANUEL ALFONSECA, Doctor Ingeniero de Telecomunicación y Licenciado en Informática
JOSE ARTECHE, Ingeniero de Telecomunicación

Diseño y maquetación:
BRAVO-LOFISH.

Dibujos:
JOSE OCHOA Y ANTONIO PERERA.

Tomo XV. **Cómo construir su propio ordenador**
AULA DE INFORMATICA APLICADA
JOAQUIN SALVACHUA, Diplomado de Telecomunicación
JOSE LUIS TODO, Diplomado de Telecomunicación
FERNANDO SUERO, Diplomado de Telecomunicación

Ediciones Siglo Cultural, S.A.

Dirección, redacción y administración:
Sor Angela de la Cruz, 24-7.º G. Teléf. 279 40 36. 28020 Madrid.

Publicidad:
Gofar Publicidad, S.A. Benito de Castro, 12 bis. 28028 Madrid.

Distribución en España:
COEDIS, S.A. Valencia, 245. Teléf. 215 70 97. 08007 Barcelona.
Delegación en Madrid: Serrano, 165. Teléf. 411 11 48.

Distribución en Ecuador: Muñoz Hnos.

Distribución en Perú: DISELPESA.

Distribución en Chile: Alfa Ltda.

Importador exclusivo Cono Sur:
CADE, S.R.L. Pasaje Sud América. 1532. Teléf.: 21 24 64.
Buenos Aires - 1.290. Argentina.

ISBN del tomo: 84-7688-053-7.
ISBN de la obra: 84-7688-018-9.

Fotocomposición:
ARTECOMP, S.A. Albarracín, 50. 28037 Madrid.

Imprime:
MATEU CROMO. Pinto (Madrid).

Depósito legal: M-1132-1987.
Printed in Spain - Impreso en España.

Suscripciones y números atrasados:
Ediciones Siglo Cultural, S.A.
Sor Angela de la Cruz, 24-7.º G. Teléf. 279 40 36. 28020 Madrid

Octubre, 1986.
P.V.P. Canarias: 365,-

INDICE

Los programas que aparecen en este libro funcionan en los ordenadores:

IBM-PC, XT, AT y compatibles.
AMSTRAD-464, 664, 6128, 1512.
SINCLAIR-SPECTRUM 48 K, 128 K, PLUS, PLUS 2.
MSX-Todos los modelos.
COMMODORE-CBM 64 y CBM 128.

PARTES DEL ORDENADOR 1

VAMOS a ver, paso a paso, cómo funciona un ordenador; de qué partes se compone y cómo se conectan entre sí. A partir de estas ideas generales veamos cómo llevarlas a la práctica con la construcción de un pequeño, pero versátil, microcomputador.

Todo ordenador tiene dos partes bien diferenciadas: el hardware y el software. El hardware es el conjunto de circuitos que componen físicamente el ordenador. El software son los programas e instrucciones que indican a los circuitos qué funciones deberán realizar.

En este capítulo describiremos brevemente el hardware. Los ordenadores tienen tres partes básicas:

— La unidad central de procesos (CPU) o microprocesador.
— La memoria.
— La unidad de entrada-salida (I/O)*.

La CPU es el «cerebro» del ordenador. Es la que controla a las demás partes y la que realiza las operaciones.

La memoria es un almacén donde están guardados los datos y las instrucciones que la CPU debe utilizar y en ella almacena los resultados.

Con esto parece que tenemos todo lo necesario, pero no es así, ya que para que los resultados de las operaciones puedan ser útiles es necesario comunicarse con el exterior. De esta función se encarga la unidad de I/O. Se encarga de transmitir datos del ordenador al exterior (pantalla del ordenador, impresora, etc.) y del exterior al ordenador (teclado, cinta cassette, etc.).

* Las iniciales corresponden a las palabras en inglés. (CPU = Central Processing Unit; I/O = Input/Output). A partir de ahora todas las iniciales se referirán a su escritura en inglés, ya que esta forma es la comúnmente aceptada.

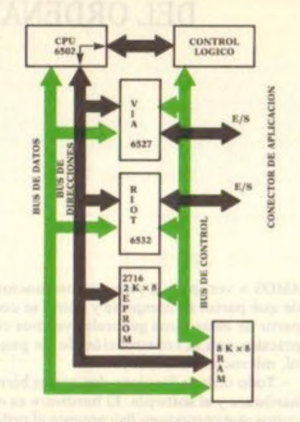


Fig. 1.1. Diagrama de bloques del ordenador.

El ordenador sólo trabaja con datos que le son comprensibles. Estos son bits, o dígitos binarios (para mayor información, ver el apéndice A). Cada bit sólo puede ser o un 1 o un 0. Pero como el ordenador no trata con números, esto se representará como 0 voltios, cuando se quiera representar un 0, y como 5 voltios cuando se quiera representar un 1. A los grupos de 8 bits se les conoce como bytes. Los microprocesadores llamados «de 8 bits» sólo pueden procesar al mismo tiempo un byte.

Como se observa en la figura 1.1, en el ordenador cada bloque está relacionado con los demás mediante una serie de líneas, o cables, conocidos como buses.

Existen tres buses: el de control, el de datos y el de direcciones.

El bus de control es el que se encarga de llevar desde la CPU a los demás bloques las instrucciones que éstos deben realizar.

El bus de datos es el que lleva la información, o datos, que los distintos bloques se comunican. El ancho de este bus (número de bits que se corresponde con el número de cables) es el de ancho de palabra = 1 byte.

El bus de direcciones da la dirección, como su nombre indica, adonde debe llevarse el dato. Está organizado como las direcciones de correos.

Por ejemplo, para que la CPU almacene un dato en la memoria debe enviar por el bus de control la señal de que quiere guardar el dato en la

memoria, el dato por el bus de datos, y en qué parte de la memoria quiere almacenarlo por el bus de direcciones. Del ancho del bus de direcciones dependerá el máximo de memoria que el ordenador puede utilizar.

Esta organización, o arquitectura, se conoce como arquitectura von Neumann, en honor del primero al que se le ocurrió. John von Neumann la aplica en los años cuarenta a la construcción de los primeros ordenadores, y desde entonces no se han realizado realmente diferentes. Actual-

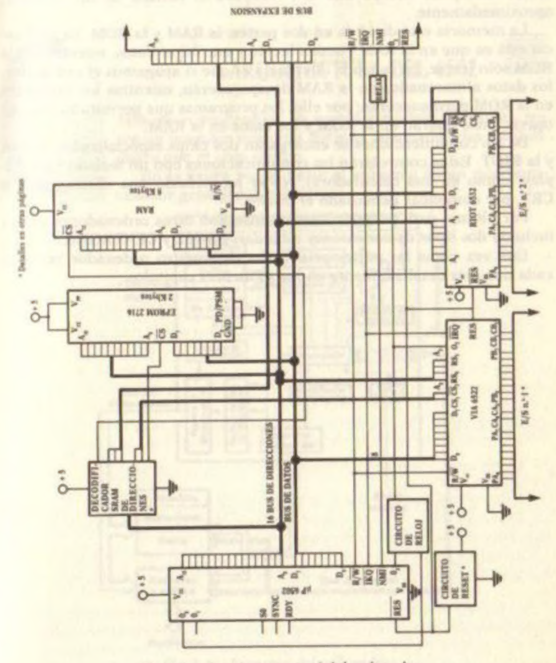


Fig. 1.2. Esquema general del ordenador.

mente se intenta buscar alguna que permita a varias CPUs realizar tareas cooperando entre ellas, pero no se ha encontrado ninguna de utilidad comparable.

Veamos ahora cómo se realizan estas ideas generales en la realización de nuestro ordenador.

Como CPU vamos a usar el microprocesador 6502. Este es un «chip», o circuito integrado, que lleva todo lo necesario para componer una CPU. Hasta hace poco tiempo una CPU ocupaba el tamaño de un televisor, aproximadamente.

La memoria está dividida en dos partes: la RAM y la ROM. La diferencia está en que en la RAM pueden leerse y escribirse datos, mientras en la ROM sólo leerse. La segunda diferencia es que si apagamos el ordenador, los datos almacenados en la RAM desaparecerán, mientras los existentes en la ROM permanecerán; por ello, los programas que permitirán realizar operaciones estarán en la ROM y los datos en la RAM.

De las comunicaciones se encargarán dos chips especializados, la VIA y la RIOT. Estos controlarán las comunicaciones con un teclado y un display (como en una calculadora). No se ha incluido una «Televisión», o CRT, por complicar demasiado el diseño.

Por último, para permitir comunicarse con otros ordenadores se han incluido dos tipos de conexiones estándar: RS-232 y CENTRONICS.

Una vez vistas las partes principales de nuestro ordenador veremos cada una más detalladamente en los siguientes capítulos.

EL MICROPROCESADOR 2

ARQUITECTURA DEL 6502

CADA microprocesador, de marca y modelo diferentes, posee un lenguaje máquina distinto, adecuado a las posibilidades que el chip ofrece.

En la figura puede verse el esquema de un microprocesador genérico.

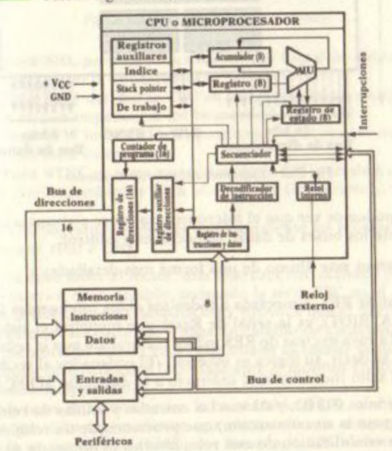


Fig. 2.1. Esquema del ordenador.

En nuestro ordenador utilizaremos el microprocesador 6502, de la marca Rockwell.

Este es un microprocesador de 8 bits, es decir, su bus de datos puede manejar simultáneamente palabras de 8 bits. Puede manejar hasta 64 Kbytes de memoria (su bus de direcciones es de 16 bits: 2^{16} bits = 65536 bytes = 64 Kbytes). Y la frecuencia de reloj es de 1 MHz (en la versión que utilizaremos). En la siguiente gráfica podemos observar la estructura interna del microprocesador 6502 desde el punto de vista del programador (arquitectura software).

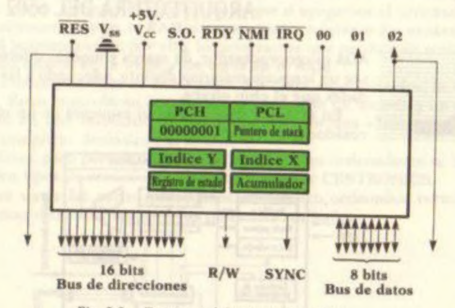


Fig. 2.2. Esquema del microprocesador.

En ella podemos ver que el microprocesador se comunica con el exterior mediante los buses de datos, direcciones y control.

Describamos este último de una forma más detallada:

— La patilla RES, conectada a todos los chips inteligentes (microprocesador, VIA, RIOT), es la señal de Reset que inicializa el sistema al encenderse. (La raya encima de RES indica claramente que la señal es activa a nivel bajo es decir, su lógica es inversa.) (El ordenador al recibir esta señal hace un salto incondicional indirecto a la dirección $FFFC $FFFD.)

— Las señales Ø1, Ø2, y Ø3 son las entradas y salidas de reloj. Ø1 y Ø2 son salidas (para la sincronización) que provienen de un reloj interno del chip. Para la estabilización de este reloj interno es necesaria una señal de reloj exterior conectada a Ø0, según uno de los esquemas de la figura.

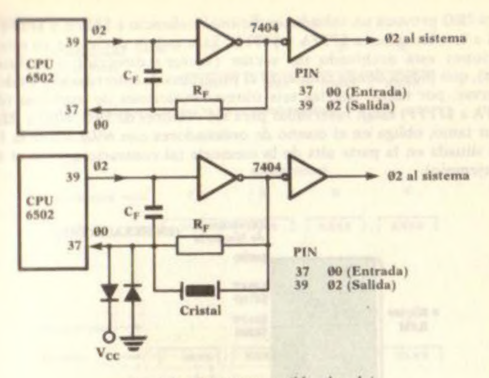


Fig. 2.3. Esquemas posibles de reloj.

— La línea S.O. permite la activación del flag V de sobrepasamiento desde el exterior (no la utilizaremos).

— La línea RDY detiene la CPU en todos los ciclos, excepto en escritura, y deja en alta impedancia los buses (no la usamos).

— La línea R/W indica a los dispositivos si el acceso es de lectura (nivel alto) o escritura (nivel bajo).

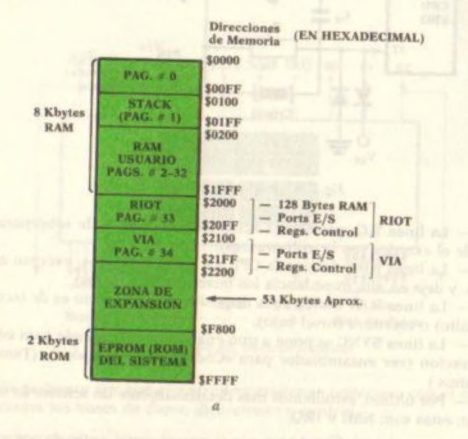
— La línea SYNC se pone a uno cuando la CPU accede a un código de operación (ver ensamblador para «Código de Operación»). (Tampoco la usamos.)

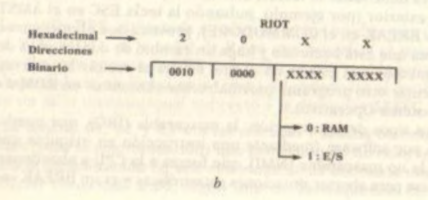
— Por último, estudiemos más detalladamente las señales de interrupción; éstas son: NMI e IRQ.

Como ya debe saber el lector, una interrupción, activada por una señal hardware exterior (por ejemplo, pulsando la tecla ESC en el AMSTRAD o en el PC o BREAK en el COMMODORE), provoca que el microprocesador deje la tarea que está haciendo y haga un cambio de contexto, es decir, archive en memoria el estado en el que estaba al recibir la interrupción y pase a ejecutar otro programa (normalmente el monitor en ROM u otra rutina del Sistema Operativo).

Hay dos tipos de interrupción; la mascarable (IRQ), que puede ser inhabilitada por software (mediante una instrucción en máquina que la desactiva) y la no mascarable (NMI), que fuerza a la CPU a abandonarlo todo (suele usarse para abortar situaciones catastróficas = es un BREAK «salvaje»).

La IRQ provoca un salto incondicional indirecto a $FFFE y $FFFF y la NMI a las direcciones $FFFA y $FFFB. Esto quiere decir que en estas direcciones está archivado un vector (vector = dirección = 16 bits = 2 bytes), que indica dónde comienza el programa de interrupción. Podemos observar, por tanto, que las seis últimas posiciones de memoria (de la $FFFA a $FFFF) están reservadas para los vectores de IRQ, NMI y RESET y, por tanto, obliga en el diseño de ordenadores con 6502 a que la ROM vaya situada en la parte alta de la memoria (al contrario que en el Z-80, por ejemplo).





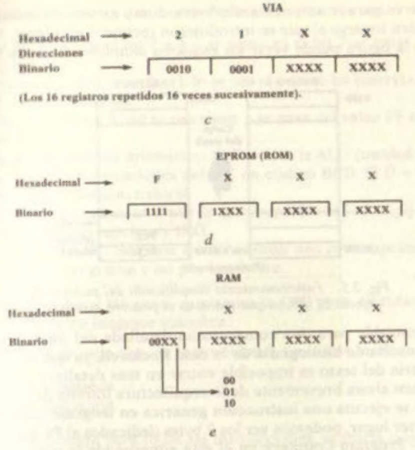


Fig. 2.4. Mapa de memoria.

Veamos ahora más detalladamente la arquitectura software del 6502.

Además de las seis posiciones anteriormente citadas, hay dos páginas de memoria (una página = 256 bytes = de $XXOO a $XXFF), que son usadas de forma especial por el 6502.

La página cero ($0000 a $00FF) es una zona de memoria de acceso rápido (ya que se ahorra un ciclo de máquina cada vez que se usa), por lo que suele utilizarse para almacenar en ella variables o datos de uso frecuente (normalmente las variables internas del sistema). La página 1 ($0100 a $01FF) es la zona reservada al STACK o pila. ¿Qué es la pila? Intentaremos explicarlo brevemente.

Cuando en el programa máquina hay que hacer un cambio de contexto (un salto a subrutina o una interrupción), hace falta archivar el estado de la CPU o enviar una serie de parámetros, para lo que se necesita una zona de memoria de acceso rápido en la que almacenar en un orden preestablecido esta información.

La pila ofrece un método muy sencillo mediante el puntero del stack (uno de los registros de la CPU accesible por software en lenguaje máquina).

Esta sirve para ir amontonando bytes, como en un «montón» y sacarlos en orden inverso al que se introdujeron (primero en entrar, último en salir). En la figura puede verse un esquema simplificado de su funcionamiento.

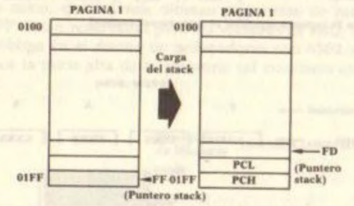


Fig. 2.5. Funcionamiento simplificado del puntero de stack. El último que entra, es el primero que sale.

Es aconsejable para una comprensión detallada del microprocesador 6502 la consulta de bibliografía de la casa Rockwell, ya que por la brevedad necesaria del texto es imposible entrar en más detalles.

Hablemos ahora brevemente de la arquitectura interna de los registros y de cómo se ejecuta una instrucción genérica en lenguaje máquina.

En primer lugar, podemos ver los 2 bytes dedicados al PC (contador de programa - Program Counter); en él está almacenada la posición de memoria en la que está el byte que estamos ejecutando.

El puntero del stack (en realidad, un solo byte, ya que el otro es fijo e indica la página 1) contiene la dirección del último byte almacenado en la pila.

Los registros de índice X e Y se utilizan en las operaciones en las que se necesitan valores de índice o desplazamiento relativo y son de 8 bits (son registros versátiles, pero no de uso general).

El acumulador es el registro primordial de la CPU. En él se realizan todas las operaciones aritméticas y lógicas.

Por último, el registro de estado es un registro de 8 bits, 7 de los cuales son flags (banderas o indicadores) que nos indican cuál es el estado de la CPU en cada momento. En la figura puede verse la colocación de cada indicador.

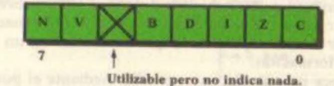


Fig. 2.6. Estructura de los bits del registro de Estado, que actúan como señalizadores o alarmas (flags).

A continuación damos una breve explicación de cada uno:

N: Flag de signo: si el bit 7 = 1 y el signo es el bit 7 → valor negativo en el Ac. y N = 1.

V: Sobrepasamiento (overflow): Si se usa el octavo bit (carry) como signo e igual a 1 → overflow.

C: Señal de acarreo: Si no se usa signo y se pasa del valor FF en el acumulador el C = 1.

D: Flag de tratamiento aritmético: Si D = 1 → la ALU (unidad aritmético-lógica) realiza las operaciones del AC en código BCD. Si D = 0 → operaciones en código binario natural.

I: Bandera de enmascaramiento de interrupción: Si I=1 implica que la CPU no aceptará interrupciones IRQ.

B: Señal de BREAK: Se pone a uno cuando una interrupción IRQ ha sido activada por programa y no por hardware.

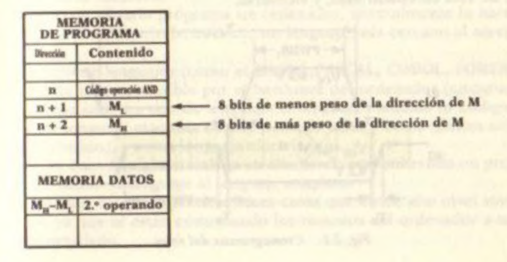
Describamos ahora los pasos que sigue la CPU para ejecutar una instrucción genérica en lenguaje máquina.

La ejecución de todo tipo de instrucciones tiene dos fases:

— Fase de búsqueda.
— Fase de ejecución.

La primera, común para todas las instrucciones, se inicia cuando en el PC está la dirección del código de la próxima instrucción a ejecutar. Esta dirección se pone en el bus de direcciones y se activa la patilla R/W̄ (con R), con lo que la memoria (ROM o RAM) pone el contenido de la dirección en el bus de datos.

Del bus de datos la CPU lee el código y lo interpreta finalizando la primera fase (ver figura 2.7).



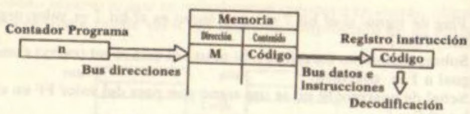


Fig. 2.7. Ejecución de una instrucción.

La fase de ejecución depende de cada instrucción. Si ésta necesita un dato, lo leerá de la memoria de forma similar a como leyó el código, almacenándolo en algún registro o usándolo según el caso. Si este dato es una dirección (2 bytes), es obvio que serán necesarios dos accesos sucesivos a memoria (ver figura).

En el apéndice se pueden ver detalladas tablas que describen todas las instrucciones, su tiempo de ejecución y otros datos de interés. Haremos una descripción breve de ellas en la parte dedicada al ensamblador.



BUSES Y CRONOGRAMAS

En nuestro ordenador todas las operaciones se realizan síncronamente; esto quiere decir que todas las partes implicadas en una operación actúan al mismo tiempo. Es igual que una orquesta en la que todos los músicos deben dar una misma nota al mismo tiempo, no esperan a oír la del compañero, ya que sería fatal. Al igual que los músicos se «sincronizan» al director de orquesta en el microprocesador todos los sistemas se sincronizan con el reloj. Reloj es una señal periódica, conocida como Ø1. Existe otra señal Ø2 que es la inversa de Ø1; cuando Ø1 está en estado alto, Ø2 está en estado bajo, y viceversa.

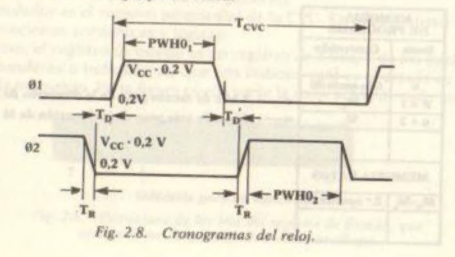


Fig. 2.8. Cronogramas del reloj.

La frecuencia de esta señal, las veces que se repite cada segundo, nos da la rapidez con que el ordenador realiza sus operaciones. En nuestro caso será de 1 MHz (un millón de veces por segundo). Aunque pueda parecer mucho, con la tecnología actual no lo es, pues existen otros microprocesadores que funcionan a 8 MHz e incluso a 16 MHz.

Como todas las señales están referidas al reloj, se suelen dibujar con respecto a Ø1 o Ø2. A partir de ahora utilizaremos algunos de estos dibujos para explicar las distintas señales de control.

El bus de direcciones está formado por 16 líneas conocidas como A1, A2, ..., A15 (la A viene de adress, dirección en inglés).

El bus de datos estará formado por ocho líneas conocidas como D0, D01, ..., D7 (la D viene de Data, es decir, dato).

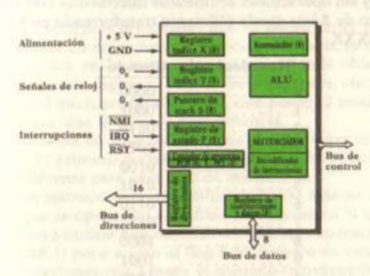


Fig. 2.9. Esquema Riot.

Los cronogramas de estos dos buses se verán en el capítulo siguiente al hablar de la memoria.

Cuando un usuario programa un ordenador, normalmente lo hace en un lenguaje de «alto nivel», es decir, un lenguaje más cercano al nivel humano.

Este tipo de lenguajes (como el BASIC, PASCAL, COBOL, FORTRAN, etc.) resulta incomprensible por el hardware del ordenador (circuitería), que sólo entiende un tipo de lenguaje: el lenguaje máquina (o código objeto). Este lenguaje máquina es una ristra de unos y ceros (pulsos altos y bajos de tensión), y es un lenguaje eléctrico.

Por eso cuando alguien trabaja en alto nivel, está utilizando un programa que traduce su lenguaje al lenguaje máquina.

El lenguaje máquina permite hacer cosas que desde alto nivel son imposibles, ya que se están controlando los recursos del ordenador a su nivel más detallado.

En contrapartida, el lenguaje máquina es prácticamente inutilizable. (Una cantidad semejante de unos y ceros volvería loco a cualquiera.)

Por ello, ya hace tiempo se creó el lenguaje ensamblador. Este lenguaje es una traducción, palabra a palabra, del lenguaje máquina; por ello, el programa que ensambla lo único que hace (en principio) es traducir una serie de códigos y números (nemotécnicos o nemónicos) a una ristra de dígitos binarios.

En el 6502 la palabra de datos tiene un ancho de 8 bits, por lo que en principio sería posible la existencia de $2^8 = 256$ instrucciones, aunque, en realidad, algunos códigos no se utilizan. Un primer paso para la utilización del lenguaje máquina es la comprensión del código hexadecimal, que permite representar números binarios de un número de bits múltiplo de 4 de forma directa y sin operaciones aritméticas intermedias (ver tabla), con lo que un número de 8 bits queda útilmente transformado en $XX y una dirección en $XXXX.

Hexadecimal	**Binario**
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

Para mayor información sobre el sistema de numeración binario, ver el apéndice.

Aunque la numeración hexadecimal simplifica enormemente la programación en máquina, carece de un sentido «natural» de número para el hombre (por ello, los ensambladores comerciales permiten la utilización de varios sistemas de numeración: hexadecimal, decimal, octal y binario).

Aunque los códigos podrían escribirse en el programa en su forma numérica, la principal ventaja del ensamblador es que traduce una palabra nemotécnica a su número correspondiente. Por eso en lenguaje ensamblador se utilizan como códigos de operación abreviaturas (del inglés) que in-

dican qué es lo que hace la instrucción. Esto (lo del inglés) obliga al programador en máquina a tener una leve idea sobre lo que algunas palabras en inglés significan. Por ejemplo, la instrucción LDA (LoaD Acumulator) significa «Carga en el acumulador el dato...» LDA es traducido por el ensamblador al número binario 1010 1001 (ver figura)

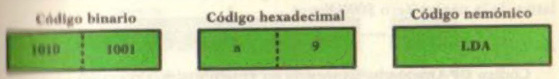


Fig. 2.10. Tres formas de expresar la instrucción de "Carga del Acumulador" mostrando la evolución del lenguaje máquina.

Uno de los conceptos principales que se deben tener en cuenta a la hora de programar en máquina es el direccionamiento, es decir, la forma en la que se indica en la instrucción cómo y dónde obtener los datos para ejecutarla (stack, registros, página cero, memoria, etc.).

En el caso del microprocesador 6502, éste posee 13 modos posibles de direccionamiento, que lo dota de gran potencia.

Las instrucciones en lenguaje máquina pueden tener una longitud de 1, 2 ó 3 bytes. El primero corresponde en todas ellas al código de operación, que es diferente para cada tipo de instrucción.

Por ello, las instrucciones que utilizan un solo byte no poseen mayor información que la operación a realizar (se supone en la mayoría de los casos que el dato a utilizar está implícito en la propia operación). Por ejemplo, CLI (CLEAR I) pone a cero el flag I del registro de estado.

En las instrucciones con 2 bytes, el segundo corresponde a un número de 8 bits (que puede ser un dato o una dirección de página cero $00XX, de Stack $01XX, etc.).

En los de 3 bytes, los dos últimos suelen ser una dirección (16 bits = 2 bytes).

Los 13 modos de direccionamiento son los siguientes:

1) Inmediato (2 bytes): El segundo byte indica el dato a utilizar (el primero es el código de operación).

Ejemplo

Código OP/Operando/Expresión en ensamblador/Expresión

A9	0 0	LDA # 00	(A)←00

2) Absoluto (3 bytes): Los dos últimos bytes indican la dirección del dato a utilizar.

Ejemplo:

Código OP/Operando/Expresión en ensamblador/Expresión

AD (2136) LDA 2136 (A)←(2136)

3) Por página cero (2 bytes): El segundo byte indica una dirección relativa de la página cero $00XX.

Ejemplo:

Código OP/Operando/Expresión en ensamblador/Expresión

A5 (00,61) LDA61 (A)←(00,61)

4) Direccionamiento por página cero, Indice X (Y) (2 bytes): La dirección del operando (que está en la página cero) se halla sumando el segundo byte de la instrucción al contenido del registro X(Y) de la CPU.

Ejemplo:

Código OP/Operando/Expresión en ensamblador/Expresión

B5 (00,38+X) LDA 38,X (A)←(00,38+X)

6) y 7) Absoluto indice X(Y) (3 bytes): La dirección donde está el dato a utilizar se halla sumando a la dirección de los dos últimos bytes de la instrucción el contenido del registro X(Y).

Ejemplo:

Código OP/Operando/Expresión en ensamblador/Expresión

BD (22,33+X) LDA 2233,X (A)←(22,33+X)

8) Direccionamiento por acumulador (1 byte): En el acumulador se encuentra el dato a utilizar.

Ejemplo:

Código OP/Operando/Expresión en ensamblador/Expresión

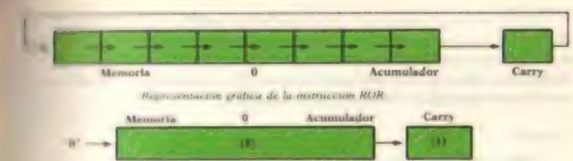
6A ROR Acumulador Ver fig. 2.11

9) Implicada (1 byte): El operando va implicado en la propia instrucción.

Ejemplo:

Código OP/Operando/Expresión en ensamblador/Expresión

18 Carry CLC C←–0



Representación gráfica de la instrucción ROR

Fig. 2.11. Representación gráfica de la instrucción LSR.

10) Relativo (2 bytes): El segundo byte de la instrucción es un valor de offset que se añade al contador de programa para provocar una bifurcación. Esta es condicional si es necesaria alguna condición (del registro de estado) para que se realice.

Ejemplo:

Código OP/Operando/Expresión en ensamblador/Expresión

F0 13 BEQ13 (PC)←(PC)+13

11) Indirecto (3 bytes): Los dos últimos bytes proporcionan una dirección. De ella y de la siguiente obtenemos otra, que es la que indica dónde está el dato (esta última dirección se suele llamar vector).

Ejemplo:

Código OP/Operando/Ensamblador/Expresión

6C 3128 JMP(3128) (PC)←–(3128) Y (3129)

12) Indexado X indirecto (primero indexa y luego indirecciona) (2 bytes): El segundo byte se añade al registro X y el resultado es una dirección de la página cero. Con su contenido y el de la siguiente obtenemos (el vector) dirección del dato.

Ejemplo:

Código OP/Operando/Ensamblador/Expresión

A1 30 LDA(30,X) (A)←–((0030)+X)

NOTA: Cuando se escribe (←—) indica el contenido de lo que hay dentro del paréntesis. Por ejemplo ($30F1), es el número $XX contenido en la dirección $30F1

13) Indirecto indexado Y (es diferente al anterior: primero hace la in-dirección y luego indexa) (2 bytes). El segundo byte indica una dirección de la página cero. Con su contenido y el del siguiente byte obtenemos otra dirección, a la que le sumamos el contenido del registro Y. La dirección resultante nos indica dónde se halla el dato.

Ejemplo

Código OP/Operando/Ensamblador/Expresión

B1	32	LDA (32),Y	(A)←–(((0032)(0033))+Y)

Veamos ahora un pequeño resumen en el que se indica brevemente la función de las principales instrucciones del 6502:

1) GRUPO DE INSTRUCCIONES ARITMETICAS Y LOGICAS

Realizan operaciones aritméticas o lógicas:

ADC	Suma aritmética
SBC	Substracción
AND	Operación lógica AND
EOR	OR exclusiva
ORA	Operación OR

2) INSTRUCCIONES DE LECTURA Y ESCRITURA EN MEMORIA

Transfieren información entre la memoria y los registros:

LDA	Cargar acumulador
LDX	Cargar X
LDY	Cargar Y
STA	Almacenar el acumulador en memoria
STX	Almacenar el registro X
STY	Almacenar Y

3) INSTRUCCIONES DE TRANSFERENCIA ENTRE REGISTROS

Intercambian la información entre registros:

TAX	Transfiere el acumulador a X
TAY	Transfiere el acumulador a Y
TXA	Transfiere X al acumulador
TYA	Transfiere Y al acumulador
TSX	Transfiere el puntero del stack a X
TXS	Transfiere el registro X al puntero del stack

4) INSTRUCCIONES PARA RUPTURA DE SECUENCIA EN EL PROGRAMA

Estas instrucciones cambian el contenido del contador de programa, lo que implica una ruptura de la secuencia normal del programa. En algunos casos esta ruptura está condicionada al contenido de algunos flags del registro de estado.

JMP	Salto incondicional
BCC	Bifurcación si C = 0
BCS	Bifurcación si C = 1
BEQ	Bifurcación si Z = 1
BNE	Bifurcación si Z = 0
BMI	Bifurcación si N = 1
BPL	Bifurcación si N = 0
BVC	Bifurcación si V = 0
BVS	Bifurcación si V = 1

5) INSTRUCCIONES DE USO DEL STACK

Manejan el puntero de stack:

PHA	Mete al acumulador del stack
PHP	Mete al registro de estado del stack
PLA	Saca al acumulador del stack
PLP	Saca al registro de estado del stack

6) INSTRUCCIONES DE TRATAMIENTOS DE SUBRUTINA

Para el uso de subrutinas e interrupciones:

JSR	Salto a subrutina
RTS	Retorno de subrutina
BRX	Provoca IRQ (interrupción mascarable) por software
RTI	Retorno de rutina de interrupción

7) INSTRUCCIONES DE CAMBIO DE LOS «FLAGS» DEL REGISTRO DE ESTADO

Ponen dichos flags a cero o unos, según el caso:

CLC	C = 0
CLD	D = 0
CLI	I = 0
CLV	V = 0
SEC	C = 1
SED	D = 1
SEI	I = 1

8) INSTRUCCIONES DE INCREMENTO Y DECREMENTO

Afectan a memoria o a registros:

DEC	Decrementa en 1 el contenido de memoria
DEX	Decrementa en 1 el registro X
DEY	Decrementa en 1 el registro Y
INC	Incrementa memoria
INX	Incrementa el registro X
INY	Incrementa el registro Y

9) INSTRUCCIONES DE TRASLADO Y ROTACION DE BITS

Permiten desplazar o rotar el acumulador (interviniendo el carry) o la memoria:

ASL	Desplaza hacia la izquierda el acumulador o memoria
LSR	Desplaza hacia la derecha el acumulador o memoria

(ver figuras)

ROR	Desplazamiento cíclico a la derecha del acumulador o memoria
ROL	Desplazamiento cíclico a la izquierda del acumulador o memoria

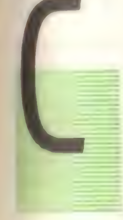
10) INSTRUCCIONES DE COMPARACION LOGICA O ARITMETICA

AND	Operación lógica AND
CMP	Compara (resta a A el contenido de M) sin afectar al acumulador. Sólo afecta al registro de estado (N Z y C)
CPX	X-M
CPY	Y-M
BIT	A AND M sólo afecta a los flags

11) INSTRUCCIONES ESPECIALES

NOP	No hace nada (pierde dos ciclos de máquina)

LA MEMORIA 3



COMO vimos en el capítulo 1, la memoria es una de las partes principales del ordenador, ya que permite tener los programas a ejecutar y los datos a usar.

Desde el punto de vista de la CPU la memoria es una serie de «casillas», cada una con una dirección, en las que se pueden leer o guardar datos. Estas direcciones son un número, siendo estas casillas consecutivas y refiriendo cada número, o dirección, a una y sólo una de las casillas.

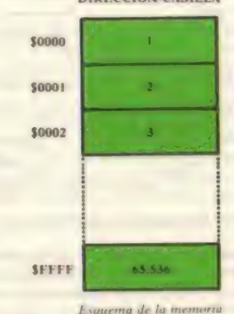


Fig. 3.1 Esquema de la memoria

La dirección se indica por un número que la CPU coloca en el bus de direcciones. Como se indicó, este número estará en notación binaria. Por ser este bus de 16 hilos, o líneas, podrá diferenciar, o direccionar, 2^{16} = 65 536 posiciones diferentes de memoria, o 64 K (por K se entiende un kilobyte, que está formado por 1024 casillas).

Los datos a leer o escribir serán colocados, o leídos, del bus de datos. Por ser este bus de datos de 8 líneas, podrán tomar los datos 2^8 = 256 valores diferentes; pudiendo representar estos datos instrucciones o datos propiamente dichos.

Por último, en el bus de control existe una señal llamada R/W (del inglés Read/Write = lectura/escritura), que indica si la CPU quiere leer o escribir un dato.

Si la señal R/W tiene un nivel lógico alto, un «1», indica que el procesador quiere leer el dato; por el contrario, si tiene un nivel lógico bajo, un «0», el procesador querrá escribir el dato.

Como se dijo en el capítulo anterior, el ordenador funciona de modo síncrono dirigido por un reloj; luego las distintas operaciones descritas anteriormente seguirán un orden en el tiempo.

Para realizar una operación de escritura la CPU coloca la dirección de la posición de memoria a leer en el bus de direcciones y un 1 en la línea de R/W durante la parte final del ciclo alto de Ø1.

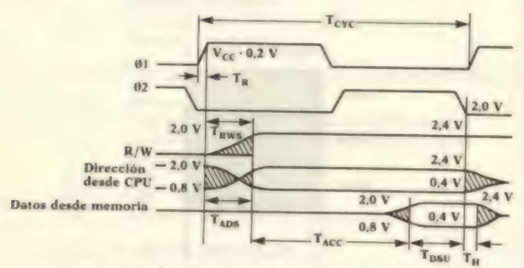


Fig. 3.2. Diagrama de tiempos de lectura.

Desde que la memoria recibe la señal de que el procesador quiere leer un dato hasta que lo coloca pasará un cierto intervalo de tiempo. En nuestro caso éste será desde que Ø2 pasa de estado bajo a estado alto, pues al final del ciclo alto de Ø2 el procesador tomará como dato el valor que la memoria coloque en el bus de datos.

El proceso de escritura es prácticamente idéntico.

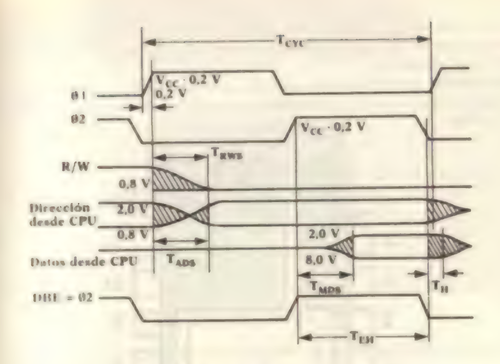


Fig. 3.3. Diagrama de tiempos de escritura.

La diferencia está en que mientras Ø2 está alto R/W está a nivel bajo y que durante la parte alta de Ø2 el microprocesador coloca el dato a escribir, es leído por la memoria y escrito.

La memoria de nuestro ordenador está formada por dos bloques principales.

La ROM (Read Only Memory = Memoria de sólo lectura) es la que almacena el programa, o software, que controla nuestro ordenador. Como su nombre indica, sólo se pueden realizar en ella operaciones de lectura. Si intentamos escribir un dato en ella, sencillamente no lo admitirá, permaneciendo la posición de memoria direccionada con el valor que tuviese en un principio. La ventaja de este tipo de memoria es que no se borra si se apaga el ordenador, es decir, los datos almacenados estarán siempre listos para la lectura.

Los circuitos usados normalmente como memoria ROM ya vienen con el programa grabado de fábrica. Por el alto coste que esto conlleva sólo se utiliza para grandes series de ordenadores comerciales, ya que así resulta más barato.

Otros circuitos de ROM son las PROM. Estas se pueden grabar, pero sólo una vez, ya que el proceso se realiza fundiendo pequeños fusibles que representan los bits. Tiene el problema que si cometemos un error éste no tendrá posible arreglo.

El chip, o circuito integrado, que utilizaremos es la EPROM (Erasable Programable Read Only Memory = Memoria programable de sólo lectu-

ra). En el es posible grabar mediante un proceso especial que consiste en intentar escribir en ella con circuitos especiales de mayor voltaje que los usados normalmente.

Es posible borrar los datos almacenados en ella iluminándola durante un largo tiempo con luz ultravioleta en una pequeña ventana existente sobre el chip.

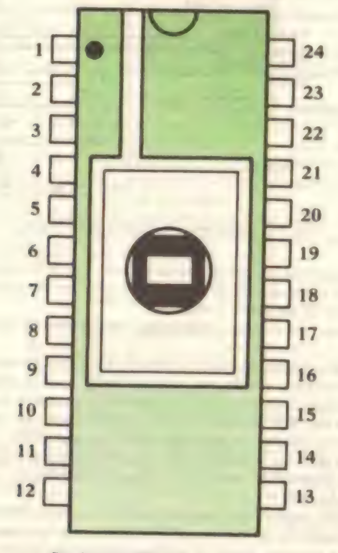
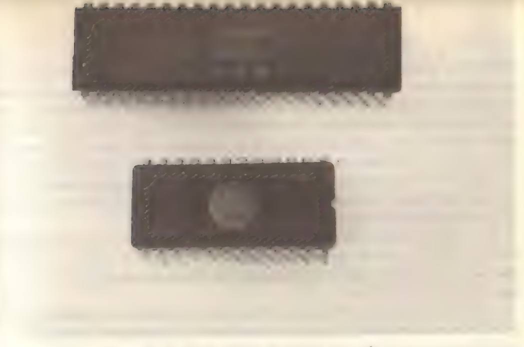


Fig. 3.4. Patillaje de una Eprom.

Por ello, esta ventana deberá estar tapada por algún tipo de etiqueta, o pegatina, ya que una excesiva exposición a los rayos solares podrían borrar nuestro programa.

El otro tipo de memoria es la RAM (Random Access Memory = Memoria de acceso aleatorio, es decir, que puede accederse a cualquiera de sus



Aspecto de la Eprom y del microprocesador.

posiciones). En esta se puede leer y escribir, pero tiene el defecto que todo lo almacenado se pierde si se apaga el ordenador.

En el comercio existen dos tipos de memoria RAM: estáticas y dinámicas.

Las estáticas son mas antiguas, pero son más sencillas de manejar por necesitar menos circuitos adicionales. Además, para sistemas pequeños como el nuestro son las más indicadas.

Las dinámicas tienen mayor capacidad, pero tienen un defecto, que olvidan los datos que tienen almacenados cada milésima de segundo, por lo que hay que estar «refrescándolas» constantemente, con la circuitería que esto lleva asociado.

Si los datos que tenemos en esta memoria quieren ser conservados deberán almacenarse en otra parte para no perderlos, tales como guardarlos en una cinta o cassette.

EL DECODIFICADOR DE DIRECCIONES

El decodificador de direcciones es una parte esencial del circuito de un ordenador. Su función es la de activar una de las patillas del circuito dependiendo de los bits de mayor peso del bus de direcciones. Es decir, de la zona de memoria en la que pretende trabajar el microprocesador y que corresponde a la posición que ocupa en ella el dispositivo y que hemos prefijado previamente mediante el diseño del mapa de memoria.

El decodificador de nuestro ordenador ha sido simplificado expresamente para que podamos realizarlo con unos pocos chips TTL-LS.

Con ellos implementamos las funciones lógicas que tomarán el valor

adecuado para activar las pastillas cuando en el bus de direcciones halla las direcciones correspondientes. Para calcular estas funciones no sólo es necesario conocimientos de electrónica digital, sino también del contenido de los chips existente en el mercado para utilizar el menor número de ellos. Existe una amplia bibliografía al respecto (ver apendice), aunque en este tipo de diseños es importante la habilidad y técnicas «artesanas».

También se puede aprovechar el hecho de que algunas pastillas tengan varias patillas o terminales de activación del chip a la hora de simplificar el circuito decodificador.

En nuestro caso el decodificador activa la pastilla de 8K de RAM (RAM estática 5565) cuando se direccionan los 8 primeros Kbytes de la memoria. Activa el RIOT cuando utilizamos las direcciones $20XX y la VIA con las direcciones $21XX.

Cuando utilizamos los dos últimos Kbytes, activa la EPROM, donde están almacenados los programas de utilidades que se proporcionan.

Nota: El único problema que se puede encontrar en el decodificador es el de los «glitches» (ver bibliografía sobre este tema). En nuestro caso, es tan sencillo y su retardo tan pequeño (20nS x n como máximo) comparado con el periodo de reloj (1 uS = 10^{-6} S).

ENTRADA-SALIDA 4



LA función de las etapas de entrada-salida es comunicar el ordenador con el exterior. En la mayoría de los casos debe adaptar el tipo de señal que le llega del exterior a una señal que sea comprensible por el ordenador.

El lugar de llegada de los datos del exterior se conoce normalmente como port, o puerto, siendo en nuestro caso de igual longitud que el bus de datos, 8 bitios.

Por ser este tipo de circuitos, usualmente, complicados y voluminosos usaremos dos circuitos integrados especializados en estas tareas: la VIA 6522 y el RIOT 6532, de la misma familia que el microprocesador que utilizamos.

La VIA (Versatil Interface Adapte = Adaptador versátil de periféricos) dispone de los siguientes elementos:

— Dos puertos, conocidos como A y B, de ocho líneas. Cada línea es programable como entrada o salida independientemente.

— Cuatro líneas de control, dos de cada puerto. Son conocidas como CA1, CA2, CB1 y CB2.

— Un registro de desplazamiento de 8 bits para convertir la información serie en paralelo, y viceversa.

— Dos temporizadores de 16 bits usados para contar o generar impulsos.

— Lógica para interrumpir a la CPU mediante la señal IRQ.

— Registros programables para el uso de los diferentes recursos.

Los registros de la VIA se sitúan en la memoria principal usando la lógica de decodificación que se vio en el capítulo anterior. Existen 16 posiciones de memoria que se direccionan por 4RS0, RS1, RS2, RS3

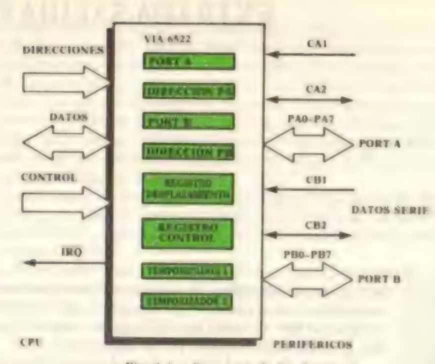


Fig. 4.1. Esquema de la via.

(4 bits → 2^4 = 16 posiciones). El decodificador activa las patillas CS1 y CS2 cuando nos referimos a las direcciones $21XX.

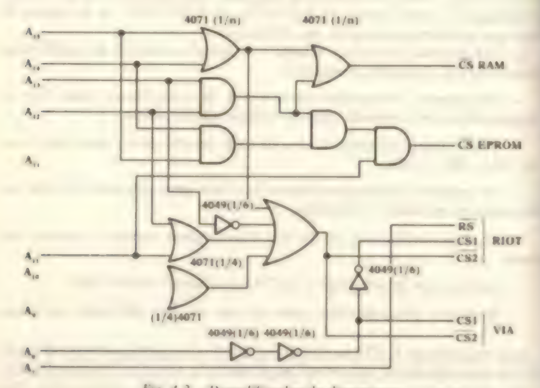
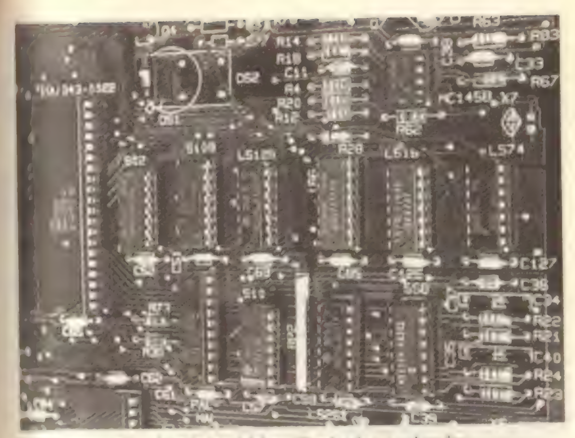


Fig. 4.2. Decodificador de direcciones.



Ejemplo práctico del decodificador de un ordenador.

Las direcciones de los port son $2100 para el B y $2101 para el A. Las direcciones de los registros de control son $2102 para el B y $2103 para el A.

Para programarlos cada bit se colocará un 0 si se quiere programar como entrada y 1 si se quiere programar como salida en los registros de direcciones.

Por ejemplo, para programar todo el port B como salida pondríamos en su registro de control de direcciones el número:

11111111

después de esto todo lo que colocáramos en los registros de los ports saldría al exterior; por el contrario, si lo que queremos es leer en el port B, en el registro de control se colocaría:

00000000

y después el valor que leeríamos del registro del port B sería el que le estuviera llegando del exterior.

Veamos dos ejemplos:

```
:L

     1          LDA  #%11111111 ;PROGRAMA SALIDAS
     2          STA  DRBV
     3          LDA  PBV         ; LEER DATOS

:
```

```
:L

     1          LDA  #%00000000 ;PROGRAMA SALIDAS
     2          STA  DRBV
     3          STA  PBV         ; ESCRIBE DATOS
```

El registro de control de las líneas CA1, CA2, CB1 y CB2 está en la posición $200C. Su configuración se observa en la figura 4.3.

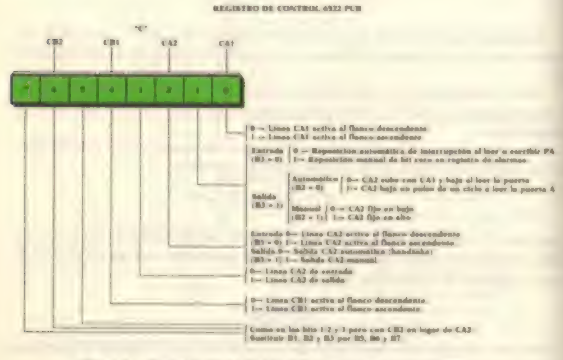


Fig. 4.3. Función de cada bit *del registro de control PCR.*

Estas líneas, dos para cada port, sirven para establecer un protocolo, o handshaking. Esta es la forma que tienen dos ordenadores para comunicarse. Antes de transmitirse datos deben mandar la señal equivalente de «que van los datos» y si los recibe bien, decir «ya los tengo». El equivalente de las personas son las frases hechas que se utilizan al comenzar una conversación; por ejemplo, dos amigos se encuentran, y antes de contarse nada de interés dicen:

Hola, Fulano, ¿la familia que tal?

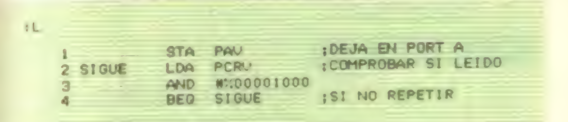
a lo que el otro contesta:

La familia bien, ¿y tú que tal?

Bien

le responde, y a partir de aquí comienzan a hablar de los temas que les interesan.

En el siguiente programa observamos un sencillo protocolo que usa sólo una señal de ocupado (Busy). Suele ser el usado para comunicarse con una impresora.



```
:L

 1              STA  PAV          ;DEJA EN PORT A
 2 SIGUE        LDA  PCRV         ;COMPROBAR SI LEIDO
 3              AND  #%00001000
 4              BEQ  SIGUE        ;SI NO REPETIR
```

La posibilidad de realizar entrada-salida por interrupciones es muy útil. Para ello existe el registro de control situado en la posición $200D el de alarma y en la $200E el de activación.

En la figura 4.4 vemos las misiones de los distintos registros.

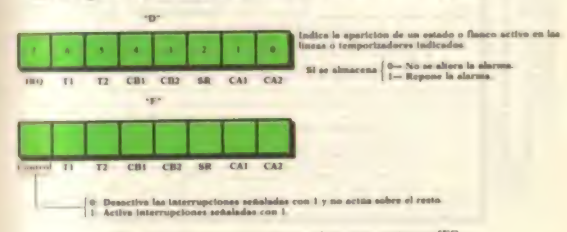


Fig. 4.4. Registro de alarmas de interrupciones IFR

La utilidad de la interrupción es clara en el siguiente ejemplo. Supon gamos que tenemos que realizar unas operaciones en la CPU, y leer, por ejemplo, un teclado. Por ser la velocidad de las personas lenta comparada con el ordenador. Si realizáramos un bucle de espera que comprobase si se ha pulsado alguna tecla se perderían algunos segundos, durante los cua les la CPU no ha realizado ninguna de las operaciones pendientes. Por tan to, si hace las operaciones, estas podrían durar demasiado tiempo y no leer la tecla cuando fuese pulsada. La solución es programar la VIA de forma que cuando llegue un dato genere una señal de interrupción, se lea la te cla pulsada y se continúe con la ejecución del programa.

La VIA también dispone de dos temporizadores, o Timers, que tienen diversas utilidades: generar cada cierto número de reloj una interrupción, cortar los impulsos que llegan por las patillas PB6 o generar dichos im pulsos por la línea PB7.

En la figura 4.5 se ve qué registros utiliza y su forma de programación

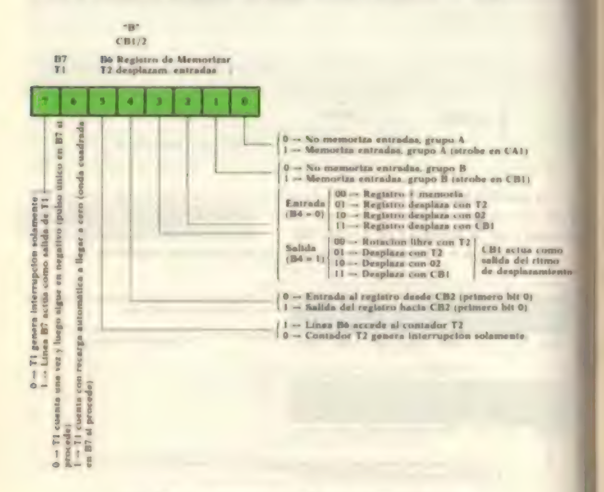


Fig. 4.5. Registro auxiliar de control ACR.

Por último, nos queda por ver el registro de desplazamiento. Este sirve para convertir los datos serie en paralelo, y viceversa.

Los datos están en paralelo, por ejemplo, en el bus de datos. Al man darse un dato se mandan los ocho bits simultáneamente por las ocho lí neas. Esto requiere menos tiempo, pero ocupa más sitio.

El enviar los datos en serie es enviarlos en «fila india». Esto ocupa más tiempo, pero sólo una línea. Para convertir de una forma a otra se utiliza el registro que pasa los bits que le llegan desplazando los que ya le han lle gado, de ahí el nombre.

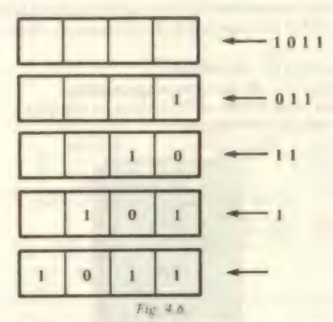


Fig. 4.6

El registro de desplazamiento está situado en la posición de memoria $200A y el control en $200B (compartido con el temporizador).

Bits de ACR			MODO DE FUNCIONAMIENTO DE SR
2	1	[illegible]	
0	0	0	Registro de desplazamiento inhabilitado
0	0	1	Entrada de datos en serie a SR bajo el control de T2
0	1	0	Entrada de datos en serie a SR bajo el control de 0_T
0	1	1	Entrada de datos en serie a SR bajo el control de un reloj externo
1	0	0	Salida de datos en intervalor continuos de T2.
1	0	1	Salida de datos en serie de SR bajo el control de T2.
1	1	0	Salida de datos en serie de SR bajo el control de 0_2
1	1	1	Salida de datos en serie de SR bajo el control de un reloj externo

Fig. 4.7. Funcionamiento del registro de desplazamiento.

La frecuencia con que se realiza el desplazamiento puede determinarse por:

— La activación de la bandera T2.
— Con 02.
— Los flancos descendentes de CB1 (por flanco descendente se entiende el paso de un nivel alto a un nivel bajo).

Un ejemplo de uso de este registro se ve mas adelante en el interfaz RS-232.

El otro circuito especializado de entrada-salida es la RIOT (RAM Input-Output Timer = RAM Entrada-Salida Temporizador). Contiene los siguientes elementos:

— Memoria RAM de 128 × 8 bits.
— Dos puertos (A y B) también programables.
— Generador programable de intervalos de tiempo y con interrupción.
— Circuito detector de flancos.

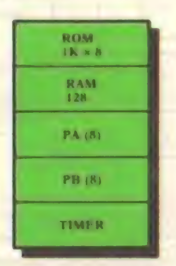


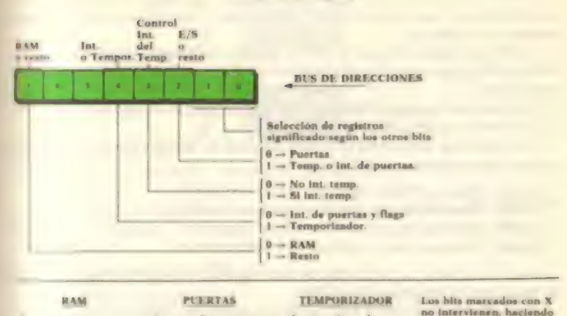
Fig. 4.8. RIOT.

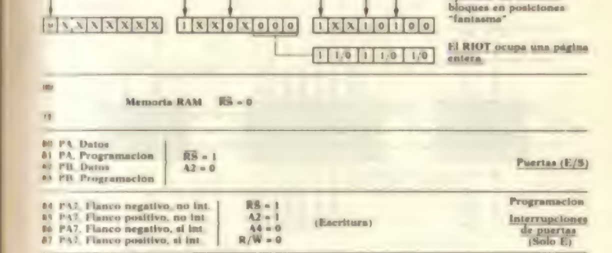
La memoria RAM es idéntica a la utilizada por el ordenador.

Sus direcciones en nuestro mapa de memoria son desde $2000 hasta $207F. El decodificador descrito en el capitulo anterior la selecciona utilizando las líneas CS1 y CS2.

La función de esta memoria es para usar el ordenador como un control industrial. Se puede realizar una configuración mínima con el microprocesador y la RIOT (versión de este mismo chip que incorpora una memoria ROM).

Los ports de la RIOT son idénticos a los vistos para la VIA. La dirección de port A es $2080 y la de su registro de control es $2081. La dirección del port B es $2082 y la de su registro de control $2083.





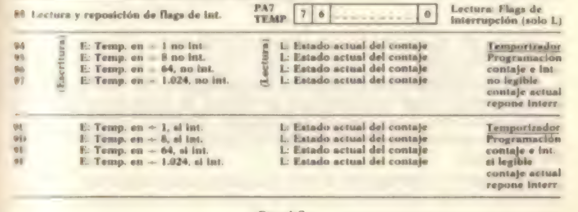


Fig. 4.9.

En este chip los ports no son exactamente iguales, en port B es capaz de suministrar una corriente de 3mA, lo que permite controlar directamente un transistor.

La última línea del port A (PA7), además de usarse como línea de entrada-salida, sirve como línea detectora de flancos, cambios de un nivel lógico a otro.

Existen dos formas de detectar los flancos: que si al detectar dicho flanco genere una interrupción, o no.

Los registros de control de dicho detector se activan mediante la escritura en las siguientes direcciones:

$2084: Detecta flanco negativo sin interrupción.
$2085: Detecta flanco negativo con interrupción.
$2086: Detecta flanco positivo sin interrupción.
$2087: Detecta flanco positivo con interrupción.

Para desactivar la detección de flancos será suficiente leer el registro de banderas de interrupción: $2085.

La parte más útil de la RIOT es el temporizador. Este tiene tres partes.

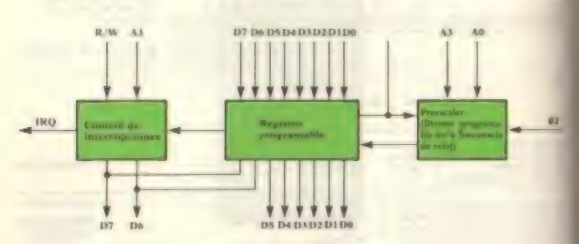


Fig. 4.10. Esquema de la Riot.

El reloj Ø2 pasa al divisor programable de la frecuencia de reloj. Este puede dividir la frecuencia por 1, 8, 64 y 1.024.

La señal dividida pasa al contador propiamente dicho. Este puede programarse para contar 255 intervalos de tiempo. Al llegar a 255 generará, si se lo indicamos, una señal de interrupción. También leen el contador, con lo que podemos tener ideas sobre el tiempo transcurrido desde que los activamos.

Un ejemplo de utilización de este contador, para realizar un contador de tiempo real de veinticuatro horas, se puede ver en el temporizador descrito en el capítulo siguiente.

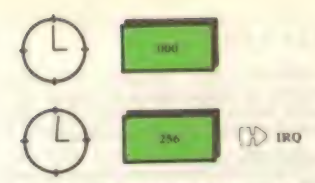


Fig. 4.11. Ejemplo del uso del contador

En la figura 4.12 se puede ver un diagrama de tiempos del funcionamiento del contador.

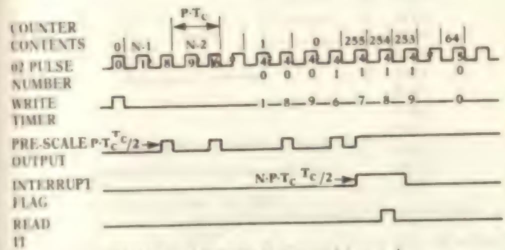
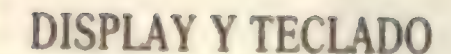


Fig. 4.12. Diagrama de tiempos del contador

APLICACIONES

Vamos a ver cómo aplicar lo que acabamos de ver a la práctica.

Como comunicaciones con las personas incluiremos un pequeño teclado de 21 teclas y un display de siete segmentos y seis dígitos. Para comunicaciones con otros ordenadores realizaremos dos interfaces: RS-232 y CENTRONICS, serie y paralelo, respectivamente. Estos son los más comúnmente utilizados en los ordenadores personales.



DISPLAY Y TECLADO

Como puede verse en la figura, utilizamos los dos ports de la vía para la lectura del teclado y la representación del display, en combinación con un decodificador de BCD a decimal. (BCD→ Binary Coded Decimal = es decir, decimal codificado en binario (ver gráfica)).

Binario	BCD gráfica
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	
1011	
1100	
1101	Valores prohibidos en código BCD
1110	
1111	

La forma de funcionamiento es la siguiente:

Con los bits del 1 al 4 del port B de la VIA seleccionamos una de las tres filas del teclado (para leer) o uno de los seis displays de siete segmentos (para escribir en ellos). (Con lo que sobra una patilla, la 3.) Al mismo tiempo utilizamos los bits del 0 al 6 del port A para leer o escribir, dependiendo del caso, las siete líneas correspondientes a los siete segmentos, o a las siete columnas del teclado. Veamos ahora por separado cada una de las dos partes: En primer lugar, veamos el proceso de lectura del teclado de una forma algo mas detallada.

Para leer el teclado programamos el PORT B como salida y activamos secuencialmente las patas, de la 1 a la 4 inclusive, escribiendo los valores BCD del 0 al 2. Con esto activamos sucesivamente una y solo una de las filas del teclado (que son tres).

Al mismo tiempo exploramos con el Port A (programado como lectura), las siete columnas con los bits del 0 al 6.

Así, cada vez que activamos una fila, leemos las siete columnas secuencialmente. Si el valor leído es 1 en algún caso, la tecla correspondiente de fila y columna estará pulsada.

Véase a continuación el programa máquina que gestiona el teclado siguiendo esta idea.

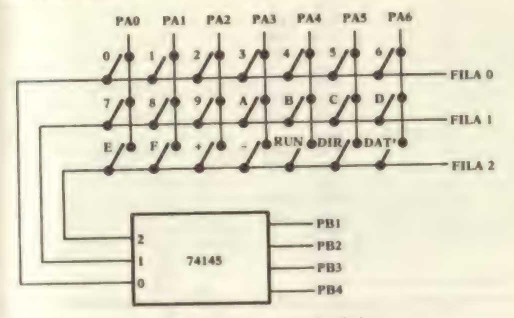


Fig. 4.13. Esquema del teclado.



```
***************************
*                         *
*     LECTURA DE UNA      *
*                         *
*        TECLA            *
*                         *
***************************
LEEKB   LDA  #$00     ;PROGRAMA PORT
        STA  PADDR
        LDX  #$06
        STX  PBDR
LEE1    LDA  PADR
        BNE  SI
        INX
        CPX  #$09
        BNE  LEE1
        JSR  ESCRIBE
        JMP  LEEKB
SI      STX  AUX1
        PHA
        JSR  ESCRIBE
        PLA
        LDX  AUX1
```

```
LEE2    LDY   #$7F
        DEY
        BPL   LEE2
        LDA   PADR
        STA   AUX
        TXA
        SEC
        SBC   #$06
        BEQ   LEFIN
LEE3    CLC
        LDA   #$06
        ADC   AUX
        DEX
        BNE   LEE3
LEFIN   LDA   AUX
        RTS
```

Hablemos ahora del display.

Debemos tener claro que cada display de siete segmentos debe ser reescrito cada cierto tiempo para que se mantenga constantemente encendido. Por ello, lo refrescaremos siempre que la CPU no tenga otra tarea más importante que hacer.

Para escribir en un determinado dígito del display programamos como salida el Port B y activamos las patas de 1 a 4 con el numero en BCD correspondiente al dígito a escribir (del 4 al 9). En el port A (programado como salida) escribimos en los bits del 0 al 7 el carácter a escribir, con lo que cada bit encenderá el segmento correspondiente.

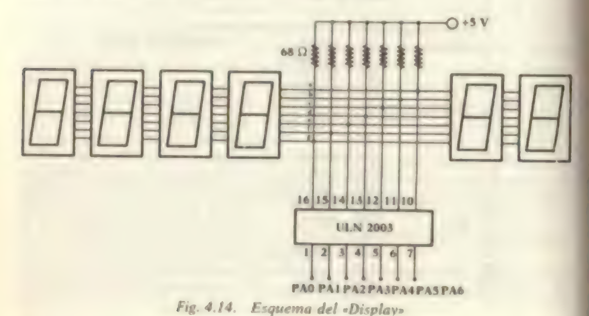


Fig. 4.14. Esquema del «Display»

Veamos el programa de control.

```
*L

 ****************************
 *                          *
 *   RUTINA DE ESCRITURA    *
 *                          *
 *      EN EL DISPLAY       *
 *                          *
 ****************************
ESCRIBE  LDA   #$7F      ;PROGRAMA PORT A
         STA   PADDR
         LDX   #$05      ;PREPARA BUFFER
ESC1     LDY   OUTBF,X   ;LEE CODIGO
         LDA   CODE,Y    ;CARGA CODIGO DISPLAY
         STA   PADR      ;ESCRIBELO
UP       LDY   #$7F      ;ESPERA UN RATO
         DEY
         BPL   UP
         STY   PADR      ;REPROGRAMA PORT A
         LDA   #$06
         STA   PBDR      ;REPROGRAMA PORT B
         DEX
         BPL   ESC1      ;FIN ?
         RTS
```

Utilizamos la tabla para encender los segmentos de cada dígito.

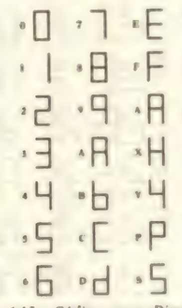


Fig. 4.15. Códigos para «Display»

Pero si quisiera usar otra, aquí damos todos los posibles códigos

Fig. 4.16. Figuras posibles del «Display».

Y una propuesta de caracteres posibles, para que el sufrido lector pueda utilizar caracteres alfanuméricos:

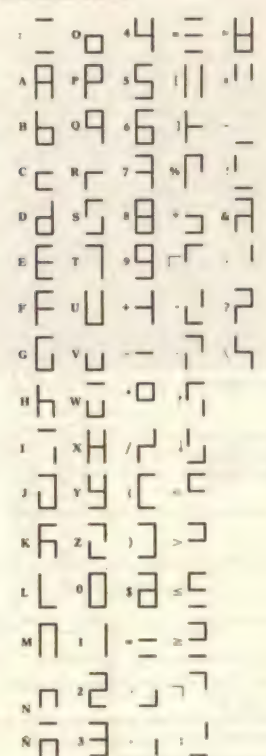


Fig. 4.17. Posible código ASCII en el «Display».



CENTRONICS

El interface paralelo más usado es el CENTRONICS. En él se comuni can los ocho bits de golpe por ocho líneas diferentes.

Uno de los datos principales de un interface es su velocidad de trans misión. Esta suele darse en baudios, que son los bits de información que transmite por segundo. Parece lógico que ésta sea lo más alta posible, pero hay que tener en cuenta que a mayor velocidad hay una mayor probabili dad que se cometan errores de transmisión (que los datos recibidos no sean los mismos que los emitidos).

Para su realización práctica usaremos el port A de la VIA para los da tos y las líneas CA1 y CA2 para el protocolo.

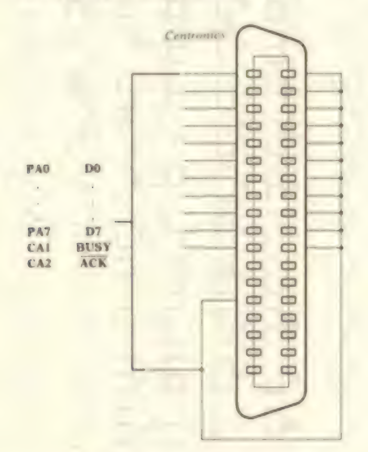
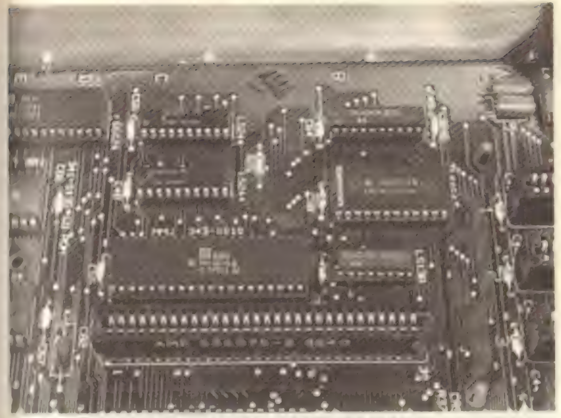


Fig. 4.18. Conexión interfaz Centronics.

Vamos a ver cómo funciona este protocolo. Este es realizado automa ticamente por la VIA, una vez programada para ello.

Vamos a ver el protocolo de lectura.

El periférico del que queremos leer debe mandar una señal por la lí-



Realización práctica de entrada/salida.

nea CA1 de que está listo para recibir los datos (si nuestro periférico no dispone de ella, deberemos tener esta señal en nivel alto). Después el puer to leerá los datos, y mandará por la línea CA2 una señal de que ha recibi do los datos.

En la siguiente figura se observa un cronograma de esta operación.

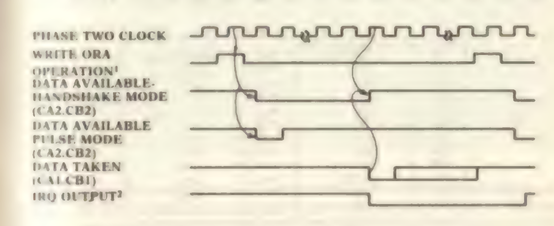


Fig. 4.19. Diagrama de tiempos del «Handshake» de escritura.

Veamos el protocolo de escritura:

Al escribir el dato a enviar en el registro del puerto éste genera la señal de dato listo en la línea CA2 y espera la señal de dato recibido en la línea CA1. Esto se observa en la siguiente figura:

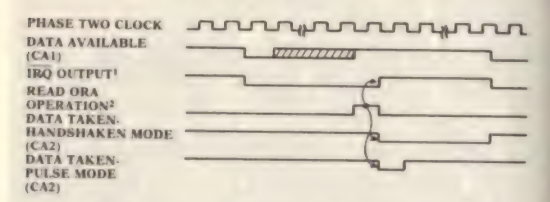
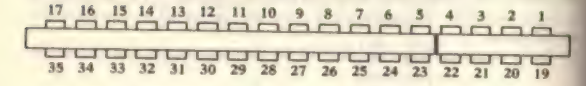


Fig. 4.20. Diagrama de tiempos del «Handshake» de lectura.

En nuestro caso el protocolo del AMSTRAD sólo dispone de la línea de «Busy» (equivalente a la señal de dato recibido vista anteriormente).

PRINTER PORT 34 WAY 0.1 EDGE CONNECTOR

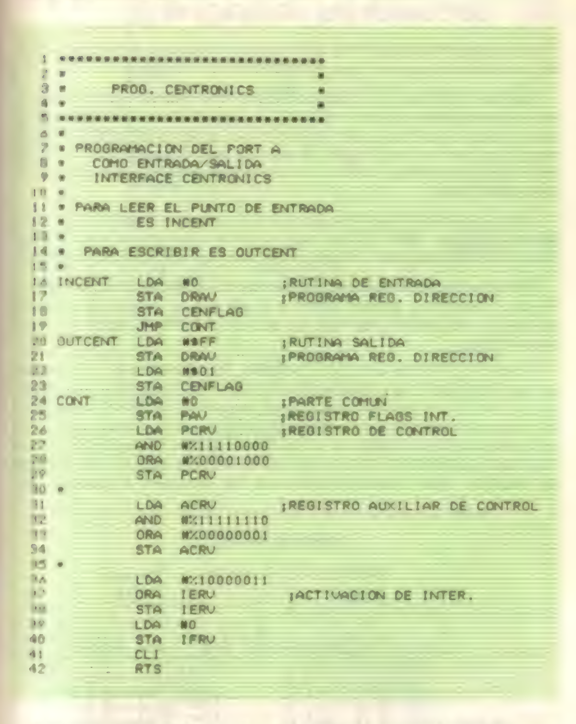
VIEWED FROM REAR



PIN 1	STROBE	PIN 19	GND
PIN 2	D0	PIN 20	GND
PIN 3	D1	PIN 21	GND
PIN 4	D2	PIN 22	GND
PIN 5	D3	PIN 23	GND
PIN 6	D4	PIN 24	GND
PIN 7	D5	PIN 25	GND
PIN 8	D6	PIN 26	GND
PIN 9	D7	PIN 27	GND
PIN 11	BUSY	PIN 28	GND
PIN 14	GND	PIN 33	GND
PIN 16	GND	All other pins	NC

Fig. 4.21. Conector paralelo Centronics.

El programa que gestiona este interfaz es el siguiente:



```
******************************
*                            *
*      PROG. CENTRONICS      *
*                            *
******************************
*
* PROGRAMACION DEL PORT A
*   COMO ENTRADA/SALIDA
*   INTERFACE CENTRONICS
*
* PARA LEER EL PUNTO DE ENTRADA
*        ES INCENT
*
*  PARA ESCRIBIR ES OUTCENT
*
INCENT   LDA  #0          ;RUTINA DE ENTRADA
         STA  DRAV        ;PROGRAMA REG. DIRECCION
         STA  CENFLAG
         JMP  CONT
OUTCENT  LDA  #$FF        ;RUTINA SALIDA
         STA  DRAV        ;PROGRAMA REG. DIRECCION
         LDA  #$01
         STA  CENFLAG
CONT     LDA  #0          ;PARTE COMUN
         STA  PAV         ;REGISTRO FLAGS INT.
         LDA  PCRV        ;REGISTRO DE CONTROL
         AND  #%11110000
         ORA  #%00001000
         STA  PCRV
*
         LDA  ACRV        ;REGISTRO AUXILIAR DE CONTROL
         AND  #%11111110
         ORA  #%00000001
         STA  ACRV
*
         LDA  #%10000011
         ORA  IERV         ;ACTIVACION DE INTER.
         STA  IERV
         LDA  #0
         STA  IFRV
         CLI
         RTS
```

Si se usase un periférico con las dos señales de protocolo, es decir, con la señal «ACK» (equivalente del papel representado por CA2 en lo visto anteriormente), el esquema de conexión sería idéntico, pero conectando CA2 a ACK.



EXPLICACION DEL INTERFACE RS-232

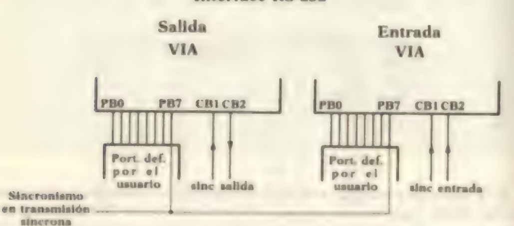
El interface RS-232 (o V24) es uno de los interfaces serie más comúnmente utilizados para la transmisión de todo tipo de datos.

Muchos periféricos de ordenador disponen del mismo, como estándar, por lo que es obvia su gran utilidad y la conveniencia de disponer de él para conseguir una mayor versatilidad en un equipo informático. Sin embargo, y tal vez por la dificultad de obtener sus tensiones, muchos de los ordenadores personales carecen de esta interface (como, por ejemplo, la gama AMSTRAD).

Nosotros hemos adaptado nuestra fuente para que proporcione las tensiones necesarias. Más adelante explicaremos cómo utilizar nuestra placa microcomputadora como adaptador de interfaces RS232-CENTRONICS.

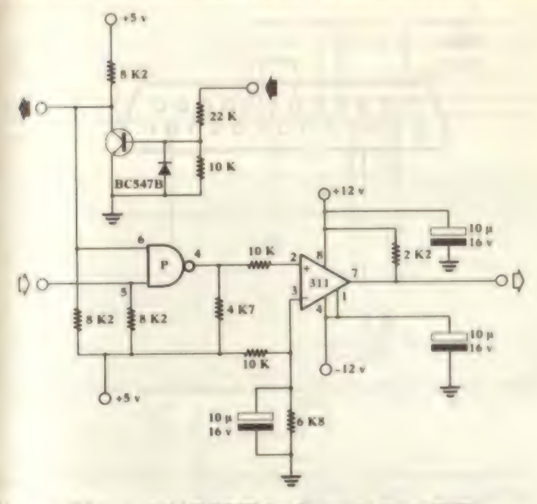
Como la transmisión es serie necesitaremos programar el Port B de la VIA adecuadamente, ya que el Port A no permite la adaptación de forma automática de paralelo (del bus de datos: ocho bits) al canal serie que necesitamos.

El RS-232 funciona con lógica negativa. Esto quiere decir que cuando queramos transmitir un «0» debemos transmitir un valor de tensión positivo (+12, aunque permite en teoría cualquiera en el rango +5÷15) y cuando deseamos transmitir un «1» negativo -12 (-5-15). Por estas razones, necesitamos hacer un cambio en el nivel de tensiones a la salida serie de la VIA, esto se puede hacer con el circuito de la figura 4.22. En ella se puede ver también unas indicaciones acerca del conector, el patillaje, etc.



a

Por cada canal de I/O necesitaremos un adaptador de nivel como el de la figura:



* La puerta P forma parte del IC. 74LS27 (que tiene cuatro puertas NAND).

b

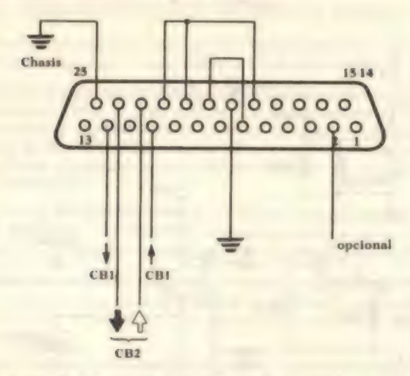
El conector RS 232/V24

1. Masa del chasis •
2. Emisión de datos ← a CB2
3. Recepción de datos
4. Solicitud de emisión •
5. Listo para emitir •
6. Bloque de datos listo •
7. Masa de señal ⊥
8. Detección de portadora

9-14. No utilizados
15. Bit de sincronización interno para emisión. a CB1
16. No utilizado
17. Bit de sincronización de recepción, procede del emisor. a CB1
18-19. No utilizados
20. Terminal de datos listo
21. No utilizado
22. Indicador de llamada entrante
23. Selector de velocidad de transmisión
24. Bit de sincronización de emisión hacia el dispositivo de transmisión (a CB1)
25. No utilizado

c

Una forma sencilla de conexión sería la siguiente:



Con este montaje (asíncrono) necesitaríamos dos cambiadores como el de la figura *d*

Fig. 4.22.

Expliquemos brevemente cómo se programa el PORT B de la VIA para la gestión serie (→ RS-232/V24).

Como entrada

	bits ACR 234		bits 76
Asíncrono	011	→ ACR:	11
Síncrono	001		

Como salida

	bits ACR 234		bits 76
Asíncrono	111	→ ACR:	11
	100		

En el caso síncrono habrá que escribir en T1 (registro de la VIA) el periodo correspondiente.

Cada vez que llegue la señal de sincronismo por CB1 se provoca IRQ, que debe estar habilitada cuando se use el interface gestionándolo por interrupción: ver el capítulo sobre la VIA.

Dependiendo de las características que el usuario desee, se debe programar el registro de control de la VIA consecuentemente. En el programa máquina de gestión del Interface se explican más detalladamente los pasos a seguir.

```
;L

*********************************
*                               *
*      PROG. RS - 232/V 24      *
*                               *
*********************************
* HAY DOS PUNTOS DE ENTRADA
* INRS PARA LA RUTINA DE ENTRADA
* Y OUTRS PARA LA DE SALIDA
*
INRS       LDA  ACRV        ;PROGRAMA EL REGISTRO AUXILIAR
           AND  #%11100001  ;DE CONTROL
           ORA  #%00001110
           STA  ACRV
           LDA  PCRV        ;IDEM REGISTRO DE CONTROL
           AND  #%00001111
           ORA  #%10000000
           STA  PCRV
           LDA  #0          ;RSFLAG=0 -> ENTRADA
           STA  RSFLAG
           JMP  CONTI
OUTRS      LDA  ACRV        ;IDEM REGISTRO AUXILIAR DE
           AND  #%11100001  ;CONTROL
           ORA  #%00011110
           STA  ACRV
           LDA  PCRV        ;IDEM REGISTRO DE CONTROL
           AND  #%00001111
           ORA  #%00000000
           STA  PCRV
           LDA  #1          ;RSFLAG=1 ->SALIDA
           STA  RSFLAG
CONTI      LDA  #0          ;ACTIVAMOS LA INTERRUPCION
           STA  IFRV        ;CORRESPONDIENTE
           LDA  IERV
           ORA  #%10000100
           STA  IERV
           CLI
           RTS

;PR#0
```

Como se observa en el programa, la velocidad de transmisión es programable por software, por lo que podrá variar según la aplicación.

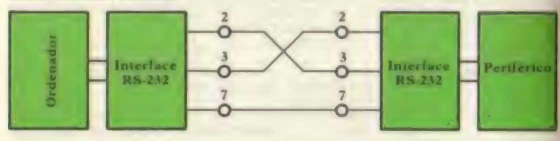


AMPLIACIONES

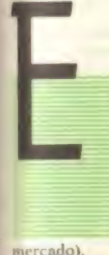
Con lo visto hasta ahora ya tenemos la versión básica de nuestro ordenador. Por sobrar líneas de los puertos de entrada salida que se pueden usar para posibles ampliaciones.

Para añadirle más memoria a nuestro ordenador será necesario colocar más decodificadores para generar la señal de selección de chips, según la cantidad de memoria que queramos añadir. Para ello, debemos observar en el mapa de memoria qué zonas están libres.

Como conexión mínima:



Una vez vista la zona en la que queremos colocar la memoria debemos ver qué direcciones, en binario, le corresponden; con ello diseñaremos un circuito combinacional que detecte dichas direcciones.



EN este capítulo trataremos de la «inteligencia» del ordenador. Con un mismo hardware se pueden realizar muchas funciones distintas, que sólo dependerán de la imaginación de quien realice los programas.

Para guardar el programa seleccionamos una EPROM capaz de almacenar 2 Kbytes (2048 bytes), por ser las más fáciles de localizar en el mercado y aunque pueda parecer extraño, son más baratas que las EPROM de 1 Kbyte (1024 bytes) (porque al ser mas antiguas escasean en el mercado).

De todas formas, siempre podremos utilizar lo que sobra de RAM para introducir programas.

Por ello, realizamos dos tipos de programas: un pequeño editor que nos permita introducir nuestros programas en código máquina, depurarlos y ejecutarlos, y un programa de reloj de veinticuatro horas, que permita ejecutar una rutina definida por el usuario al llegar una hora predeterminada.

Junto a estos programas existen diversas rutinas, que éstos utilizan, que gestionan los recursos del sistema, tales como el teclado, el display, los interfaces serie y paralelo.

Siempre podremos poner en la RAM libre nuestras propias aplicaciones, además, la mitad de la EPROM está vacía. En ella podremos almacenar nuestras utilidades.



EDITOR

La parte principal de este programa está realizado por un bucle que espera que se pulse alguna tecla.

Si la tecla pulsada es de alguna de las funciones, lo que detecta por ser su código mayor que $F, ejecuta la rutina.

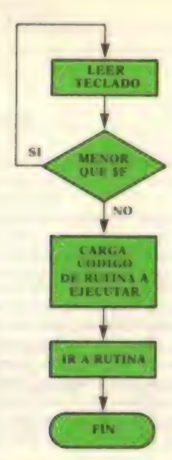


Fig. 5.1. Organigrama del bucle principal.

Para la ejecución de dicha función toma la dirección de una tabla, indexándola con el código de dicha función.

Almacena dicho dato en FUNC y ejecuta un salto indirecto a dicha posición, con lo que la dirección efectiva es la que almacenamos anteriormente.

```
*****************************
*                           *
*       BUCLE PRINCIPAL     *
*                           *
*****************************
ENTRADA JSR  LDAT
        JSR  LEEKB      ;LEE TECLA
ENT1    CMP  #$10       ;SI ES UN NUMERO
        BMI  ENTRADA    ;IGNORALO
        CMP  #$14       ;SI ERROR
        BPL  ENTRADA    ;IGNORALO
        SEC
```



```
        SBC  #$10       ;TRANSFORMAR TECLA
        ASL             ;EN VALOR
        TAX             ;PARA CALCULAR DIRECCION
        LDA  TABLA,X    ;DE LA FUNCION
        STA  FUNC
        INX
        LDA  TABLA,X
        STA  FUNC+1
        JMP  (FUNC)     ;EJECUTA LA FUNCION
```

Veamos ahora las distintas funciones del editor.

Para observar el contenido de una posición de memoria pulsaremos la tecla de función DIR, seguida de cuatro cifras hexadecimales, que serán la dirección a observar.

En el display aparecerá la dirección pulsada y el dato contenido en ella, en hexadecimal.

Si queremos observar el contenido de la dirección de memoria siguiente bastará con pulsar la tecla +.

Para observar el contenido de la anterior posicion de memoria pulsaremos la tecla -.

Si ahora queremos modificar la posición de memoria cuyo contenido estamos observando bastará con pulsar la tecla de función DATO y dos cifras hexadecimales que representarán el nuevo contenido de dicha posición de memoria.

Con esto podemos observar y modificar cualquier posición de memoria de nuestro ordenador, pero ¿cómo ejecutar un programa?

Para ejecutar un programa pulsaremos DIR y la dirección en que comienza el programa. Después pulsaremos RUN y dicho programa comenzará a ejecutarse.

Además de observar las posiciones de memoria podríamos querer observar y modificar los registros internos de la CPU.

Para observar el contenido de cualquier registro pulsaremos, sucesivamente, las teclas DIR, DATO y una tercera que representará el código del registro a visualizar. Los códigos son:

- A para el acumulador
- 0 para el registro X
- 1 para el registro Y
- 2 para el registro de status
- 3 para el puntero del stack

Una vez realizada la anterior función si queremos modificar alguno de estos registros bastará con pulsar la tecla DATO y las dos cifras hexadecimales que representen el nuevo contenido.

Con lo visto hasta ahora si queremos escribir nuestros propios programas deberemos realizar las siguientes funciones:

— Escribir el programa en lenguaje ensamblador.
— Traducir, mediante la tabla que se da en uno de los apéndices, las instrucciones por su correspondiente código hexadecimal.

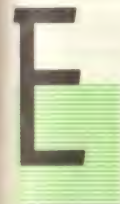
Pulsaremos la tecla DIR seguida de la dirección en que comienza nuestro programa.
Pulsaremos DATO y el código de la primera instrucción.
Pulsaremos +, DATO y el código siguiente. Esta última operación la repetiremos hasta completar el programa.
Pulsaremos la tecla DIR y la dirección de comienzo de nuestro programa y con la tecla RUN lo ejecutaremos.
Por último, para detener la ejecución de un programa y regresar al monitor, pulsaremos RESET.



TEMPORIZADOR

Es otra rutina del monitor. Programa un reloj de veinticuatro horas.
Para ejecutarlo debemos pulsar DIR, la dirección $FF2F y la tecla RUN.
Tras esta operación introduciremos la hora actual con ocho dígitos (dos para la hora, dos para los minutos y dos para los segundos). Y otros ocho para la hora en que queramos que se ejecute una rutina, cuya dirección colocaremos previamente en las posiciones de memoria $28 y $29.
Tras esto, el reloj entrará en funcionamiento.

APLICACIONES 6



EN este capítulo veremos algunos de los posibles usos del ordenador anteriormente desarrollado. Estos son algunos de los muchos posibles, ya que las posibilidades son prácticamente ilimitadas.
Los casos aquí estudiados son: un adaptador del interface RS-232 a CENTRONICS, y viceversa, y un temporizador para el control de algún electrodoméstico.

ADAPTADOR RS-232-CENTRONICS (Y VICEVERSA)

Esta aplicación pretende proporcionar al usuario de un ordenador personal que posea uno de los dos interfaces, pero carezca del otro, la posibilidad de utilizar ambos, lo que le permitirá conectar su ordenador a otros periféricos sin un desembolso excesivo.
Habrá, por tanto, dos maneras de utilizar la placa microcomputadora como interface:

1. Entrada RS-232 → Salida CENTRONICS
2. Entrada CENTRONICS → Salida RS-232

NECESIDADES HARDWARE

Hace falta para ambas el adaptador (o adaptadores) de nivel explicados en el apartado del interface RS-232/V24, así como ambos conectores estandar RS-232 y CENTRONICS.



NECESIDADES SOFTWARE

Proponemos el siguiente programa máquina para la gestión de los interfaces en esta aplicación. Es importante seguir paso a paso las explicaciones al margen para comprender con detalle los pasos a seguir.

El usuario podrá siempre cambiar el programa para adaptarlo a sus propias necesidades.

Para la primera aplicación:

```
:L

 *PROGRAMA MISMOS BUFFER
 *PARA LA ENTRADA Y SALIDA
           LDA  #$00
           STA  CENBUFP
           STA  RSBUFP
           LDA  #$08
           STA  CENBUFP+1
           STA  RSBUFP+1
           JSR  CENOUT      ;PROGRAMA CENTRONIS SALIDA
           JSR  RSIN        ;PROGRAMA RS-232 ENTRADA
           RTS              ; LO HACE

:
```

Para la segunda:

```
0

:L

 *PROGRAMA MISMOS BUFFER
 *PARA LA ENTRADA Y SALIDA
           LDA  #$00
           STA  CENBUFP
           STA  RSBUFP
           LDA  #$08
           STA  CENBUFP+1
           STA  RSBUFP+1
           JSR  CENIN       ;PROGRAMA CENTRONICS ENTRADA
           JSR  RSOUT       ;PROGRAMA RS-232 SALIDA
           RTS              ; LO HACE

:
```


En cualquier ordenador se mandarán los datos igual que se escribe por la impresora.

TEMPORIZADOR

Como ejemplo mas típico damos un controlador programable de un electrodoméstico.

Programando el temporizador de la RIOT para realizar un reloj de veinticuatro horas, podemos saber qué hora es.

Cuando llegue la hora asignada a través de un relé, interruptor controlado por tensión, podemos enchufar y desenchufar el aparato.

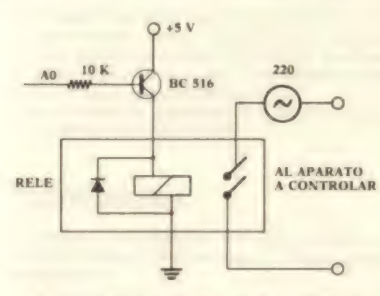
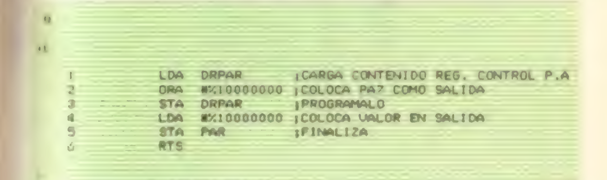


Fig. 6.1. Controlador utilizado por el temporizador.

Aquí se observa el programa encargado de ello:



```
0

:L

           LDA  DRPAR       ;CARGA CONTENIDO REG. CONTROL P.A
           ORA  #%10000000  ;COLOCA PA7 COMO SALIDA
           STA  DRPAR       ;PROGRAMALO
           LDA  #%10000000  ;COLOCA VALOR EN SALIDA
           STA  PAR         ;FINALIZA
           RTS
```

APENDICE A SISTEMAS DE NUMERACION Y SU REPRESENTACION EN EL ORDENADOR

La notación binaria (base 2) es otra forma de expresar los valores de los números. Nuestro sistema usual, el decimal (base 10) usa la combinación de diez dígitos, del cero al nueve. Los números escritos en notación binaria usan sólo dos dígitos, cero y uno. Cada posición ocupada por un dígito binario (un 0 o un 1) se llama bitio (o bit).

Internamente el ordenador representa el 1 por un valor de 5 V (valor alto) y el 0 por un voltaje nulo (valor bajo).

NOTACION DECIMAL

En notación decimal cada dígito en número representa una potencia de 10. Por ejemplo, el número 2408 en notación decimal puede escribirse en forma expandida, así:

$$(2 \times 10^3) + (4 \times 10^2) + (0 \times 10^1) + (8 \times 10^0)$$

Lo que es igual a 2408, como puede verse a continuación:

$$\begin{array}{r} 2 \times 10^3 = 2 \times 1000 = 2000 \\ 4 \times 10^2 = 4 \times 100 = 400 \\ 0 \times 10^1 = 0 \times 10 = 0 \\ 8 \times 10^0 = 8 \times 1 = 8 \\ \hline 2408 \end{array} +$$

(Obvio.)



NOTACION BINARIA

En notación binaria la que usa el ordenador internamente, cada dígito representa una potencia de 2. Por ejemplo, el número binario 101101 puede escribirse como:

$$(1 \times 2^5) + (0 \times 2^4) + (1 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0)$$

A continuación están las sucesivas potencias de 2 y su valor decimal

2^{14}	2^{13}	2^{12}	...	2^3	2^2	2^1	2^0
16384	8192	4096	...	8	4	2	1

El equivalente decimal de 101101 puede calcularse así:

$$1 \times 2^5 = 1 \times 32 = 32$$

$$\begin{aligned} 1 \times 2^4 &= 0 \times 16 = 0 \\ 1 \times 2^3 &= 1 \times 8 = 8 \\ 1 \times 2^2 &= 1 \times 4 = 4 \quad + \\ 1 \times 2^1 &= 1 \times 2 = 2 \\ 1 \times 2^0 &= 1 \times 1 = 1 \\ \hline & \qquad\qquad 45 \end{aligned}$$

OPERACIONES LOGICAS

Las operaciones lógicas se hacen bit a bit:
Las reglas para las cuatro operaciones lógicas se dan a continuación

Operador	**Regla**
AND	Si los dos bits son 1, el resultado es 1.
	Si algún bit es 0, el resultado es 0.
OR	Si algún bit es 1, el resultado es 1.
	Si los dos bits son 0, el resultado es 0.
XOR	Si algún bit, pero no los dos, es 1, el resultado es 1.
	Si los dos bits son 0, el resultado es 0.
NOT	Si el bit es 1, el resultado es 0.
	Si el bit es 0, el resultado es 1.

Ejemplo:

Cuando se realizan operaciones binarias con los números 77 y 67, estos son primero convertidos a notación binaria. El número 77 se representa en 16 bits, como 0000000001001101 y el número 67 se representa en 16 bits, como 0000000001000001. El resultado de hacer AND, OR y XOR con los dos valores es el siguiente:

	0000.0000.0100.1101
	0000.0000.0100.0001
AND:	0000.0000.0100.0001
OR:	0000.0000.0100.1101
XOR:	0000.0000.0000.1100

La resta de dos números binarios se hace sumando el primero al complementario del segundo.

Para obtener el complemento a 2 de un número binario, se cambia cada 1 por un 0 y cada 0 por 1. Entonces se suma 1 al número obtenido. Por ejemplo, el complemento a 2 de 77 se obtiene como se indica a continuación:

77 en binario	0000000001001101
Cambiando bits	1111111110110010
Sumando 1	1
-77 en binario	1111111110110011

El bit más a la izquierda = 1 significa número negativo.
negativo.

Para una descripción más detallada de la aritmética binaria búsquese en un libro sobre la materia.

Ejemplo:

Para convertir un número de notación decimal a binaria se reduce progresivamente el número decimal por la potencia de 2 mayor que no sobrepase al número hasta que éste sea nulo.

El número decimal 77 puede convertirse a notación binaria usando la técnica siguiente.

La potencia de 2 mayor que contiene el número 77 es 64 (2^6). Ponemos un 1 en esa posición del número binario, como se muestra a continuación:

128	64	32	16	8	4	2	1
0	1	0	0	0	0	0	0

Restamos a 77 el 64, con lo que tenemos 13. La potencia mayor de 2 que contiene 13 es 8 2^3), con lo que colocamos un 1 en esa posición. Restamos a 13 el 8 y queda 5. La potencia de 2 mayor que contiene a 5 es 4 (2^2) y colocamos un 1 en su lugar, restamos 5 a 4 y queda 1, con lo que colocamos un 1 en la posición de 2^0.

El número 77 en notación binaria resulta:

128	64	32	16	8	4	2	1
0	1	0	0	1	1	0	1

Se puede comprobar la exactitud de la conversión así:

$$0.2^7 + 1.2^6 + 0.2^5 + 0.2^4 + 1.2^3\ 1.2^2 + 0.2^1 + 1.2^0 = 77$$



NOTACION HEXADECIMAL

En notación hexadecimal existen 16 posibles numeros. Como solo existen 10 números (0-9), los siguientes números se representan mediante letras. Casualmente un número hexadecimal se representa por cuatro digitos binarios, por lo que esta notación se utiliza para acortar la representación de cantidades binarias.

Binario		Hexadecimal		Decimal
0000	......	0	......	0
0001	......	1	......	1
0010	......	2	......	2
0011	......	3	......	3
0100	......	4	......	4
0101	......	5	......	5
0110	......	6	......	6
0111	......	7	......	7
1000	......	8	......	8
1001	......	9	......	9
1010	......	A	......	10
1011	......	B	......	11
1100	......	C	......	12
1101	......	D	......	13
1110	......	E	......	14
1111	......	F	......	15

Por ello, un número binario de 8 bits se representará como dos digitos hexadecimales:

11110000 = F0

Para realizar la conversión de decimal a hexadecimal la cosa se complica un poco, ya que debemos ir dividiendo por 16 y tomando los restos; pero como esto es un poco complicado, recomendamos pasar previamente a binario.

La forma de sumar dos números hexadecimales es igual a la utilizada en decimal. Veamos un ejemplo:

```
  3A    A + 2 = B + 1 = C
+ BC    B + 3 = C + 2 = D + 1 = E
  --
  EC
```

La resta también sería igual.

Queda por indicar qué para indicar la base en que se representa cada número existe una notación usada por todos los programas ensambladores:

- — **Decimal: Se escribe tal cual: 255.**
- — Binario: Se antecede de %: %1111111.
- — **Hexadecimal: Se antecede de $: $FF.**

APENDICE B
CONSEJOS A LA HORA DE MONTAR EL CIRCUITO

No se debe olvidar, en primer lugar, que todo circuito integrado debe tener sus patillas de alimentación conectadas a la fuente, aunque no conste en alguna de las figuras de los circuitos. Para ello debe estudiarse detenidamente el patillaje de los mismos del apéndice. Si realizamos ampliaciones por nuestra cuenta, debemos sumar el consumo en mA de todos los chips para comprobar que nuestra fuente es suficiente. En caso contrario, deberíamos sobredimensionarla.

También es aconsejable realizar primeramente un prototipo en esas placas de pinchar componentes (pinchómetro, según los entendidos), fáciles de encontrar en cualquier tienda del ramo y que evitan la engorrosa tarea de desoldar para hacer modificaciones.

Si el lector tiene algo más de práctica en el cacharreo electrónico, puede montar su prototipo, que, si funciona, probablemente haga definitivo en alguna de las placas con pistas pregrabadas y agujereadas que se venden en el mercado. El número de cablecillos será grande, pero si el montaje se hace con orden y ajustando la longitud de los mismos, el resultado puede ser aceptable. (Aunque siempre habrá el riesgo de las capacidades entre cablecillos, que provocarán efectos indeseados.)

Si la cosa se hace más en serio, y se pretende hacer una placa de circuito impreso «elegante», es aconsejable recurrir al tablero de dibujo o utilizar un programa de diseño de los que ya existen para ordenadores personales (Smartwork, AutoCad, etc., para PC y compatibles, por ejemplo). De todas formas, para más información sobre las técnicas al uso es conveniente la consulta de bibliografía sobre el tema, imposible de tratar en un libro de estas dimensiones.

Si se sueldan circuitos integrados directamente, aunque es totalmente aconsejable el uso de zócalos (que, además, facilita la sustitución en caso de fallos), no deben calentarse más que lo que el tacto de cada uno pueda soportar. (Es una buena regla.)

Otro tema importante es el de los conectores. Cuanto mejores, más caros, pero más fiables. En nuestro caso no es necesario demasiado desembolso, pero, eso sí, deben de soldarse bien (algo difícil). Es evidente que un conector estándar, por ejemplo el RS-232/V24, no tiene más remedio que cumplir la norma, por lo que será caro. (Relativamente.)

Se aconseja el uso de conectores entre placas si se desea hacer el ordenador de forma modular. Esto permite que con una misma arquitectura básica podamos usar nuestro ordenador para muchas cosas. Por ejemplo, si ponemos un conector de placa a la salida de la VIA y es RIOT, podremos cambiar el tipo de periféricos con sólo cambiar de placa adaptadora y de programa de gestión de los controladores de periféricos, según el caso.

Los dos principales problemas a la hora de la realización práctica de un circuito digital, con los que suele toparse todo el mundo, son el acoplo y los gliches. Estos dos fantasmas que siempre acechan se pueden evitar. El primero, utilizando pequeños condensadores de desacoplo entre las salidas de los chips digitales de alimentación y masa (se aconseja el valor de unos 100 nF). El segundo, realizando un diseño adecuado. En nuestro caso, la velocidad de reloj es lo suficientemente lenta (1 MHz) para que carezca de importancia.

Cuando vayamos a soldar, debemos utilizar un soldador eléctrico de baja potencia (aproximadamente 15 W), para no dañar el circuito, y estaño de uso electrónico. (Se vende en rollos y lleva núcleo de resina para facilitar su fundición.)

Si se utilizan cables, deben utilizarse siempre lo más cortos posible, para evitar capacidades parásitas.

En el mercado electrónico de componentes los precios son enormemente variables, pudiendo cambiar en casi un 100%, dependiendo del establecimiento. Por ello, es aconsejable ir a varias tiendas antes de hacernos con el material que necesitemos.

Hablemos de la EPROM. Como ya habíamos dicho anteriormente, la EPROM es una memoria de sólo lectura (ROM) que permite ser reprogramada. Esto quiere decir que existen en el mercado unos aparatos especiales para grabarlas. Está claro que sólo es aconsejable comprar (o hacerlo, para los hábiles) un grabador de EPROM, si se graban muchas.

En nuestro caso, que sólo vamos a grabar una, esto sería superfluo. Por ello, debemos buscar algún establecimiento del ramo en el que nos la graben a partir del listado hexadecimal. (Ver apéndice E.)

En algunas tiendas, si compramos en ellas la EPROM, la programan gratis (en Madrid).

Si el lector es de provincias, y está muy desesperado, puede buscar en alguna revista electrónica algún artículo que describa cómo montar el grabador. (Se aconseja experiencia para intentarlo.)

Los chips de la familia CMOS son muy sensibles a las corrientes estáticas (al contrario que los TTL). Esto quiere decir que si tocamos las patillas (terminales) con los dedos, pueden deteriorarse. Por eso suelen venderse en tubos de plástico antiestáticos o envueltos en papel de aluminio. Cuando los vayamos a montar en su zócalo debemos agarrarlos por el cuerpo.

Aconsejamos que antes de alimentar el circuito se realice un repaso concienzudo del montaje para localizar posibles fallos:

— Soldaduras frías: Parece soldado, pero no hace buen contacto eléctrico.
— Cortocircuitos: Entre pistas por gotas de soldadura, etc.
— Circuitos abiertos: Pistas cortadas, etc.
— Mala orientación al enchufar los C.I. (pueden estropearse).
— Otras múltiples razones por las cuales no puede funcionar:

- Aparatos eléctricos cerca (interferencias)
- Falta de alimentación de algún integrado.
- Etc.

Por último, para los que vayan a diseñarse su propio circuito impreso a partir del circuito teórico que proporcionamos, les aconsejamos utilizar métodos fotográficos para realizar la placa (resina fotosensible, copia de papel vegetal, etc.).

Deseamos suerte y aconsejamos paciencia a todos.

APENDICE C
FUENTE DE ALIMENTACION

Aún no nos habíamos ocupado de alimentar a nuestro ordenador. Para que los circuitos integrados funcionen deben «enchufarse». La mayoría de ellos van conectados a 5 voltios, aunque los relacionados con el RS-232 irán conectados a ± 12 voltios. Para ello diseñaremos una fuente de alimentación que de los 220 V alterna de la red nos dé 5 y ± 12 voltios de corriente continua.

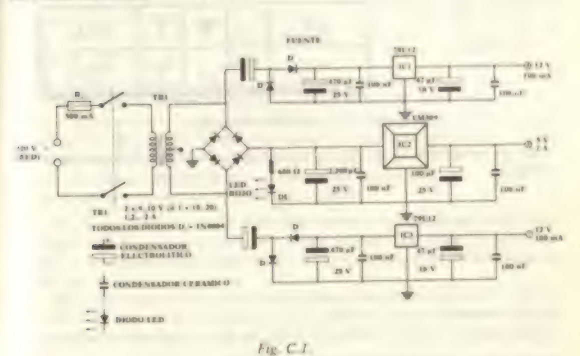


Fig. C.1.

La fuente de 5 voltios proporciona una corriente máxima de 1 amperio, la de ± 12 sólo 100 mA.

Se ha incluido un led (diodo emisor de luz) que estará encendido si la fuente funciona adecuadamente.

No debe olvidarse el colocar un disipador de potencia al circuito integrado LM 309 (CI2), pues de lo contrario éste se sobrecalentaría demasiado, pudiendo llegar a destruirse.

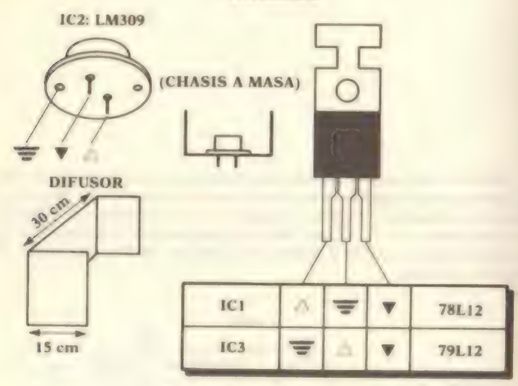


Fig. C.2. Patillajes integrados

En el apéndice D se dan los patillajes de todos los componentes utilizados.

APENDICE D
LISTADO FUENTE DEL PROGRAMA MONITOR

Aquí presentamos el listado fuente del programa monitor.

Para los no habituados a este tipo de listados en lenguaje ensamblador diremos lo siguiente:

Las líneas tienen el siguiente formato:

Listado hexadecimal	N.º línea (tres dígitos)	Listado en ensamblador o comentarios
XXXX : XX XX XX		

Los cuatro primeros dígitos (en hexadecimal) indican la dirección a partir de la cual se deben almacenar los números de dos dígitos siguientes, que pueden ser 1, 2 ó 3, dependiendo de la longitud de la instrucción ensamblador. (No se olvide que dos dígitos hexadecimales $XX = 1 byte).

El número de línea indica solamente el orden, en el listado y no es relevante de cara a lenguaje máquina (segundo campo).

En el tercer campo puede haber dos cosas:

— Si en el primero existía algo (números en hexadecimal), está claro que hay una instrucción en lenguaje ensamblador (ver tema II) y después de ella en el espacio que sobra de la línea puede haber un comentario separado por el «;».

— En caso contrario, comenzará la línea con un asterisco «*» que indica que toda ella es un comentario (o un título) y no es relevante de cara al ensamblado (aunque puede ser muy importante para el programador).

En la zona dedicada a la instrucción en ensamblador hay tres campos:

— El primero (vacío o no) es el campo para la etiqueta (nombre de variable de dirección = 2 bytes).

— El segundo es el dedicado al nemónico (Código de Operación).

— El tercero es el dedicado al operando, que puede ser:

- Una etiqueta (16 bits = 2 bytes = dirección).
- Una variable (8 bits = 1 byte = dirección o dato).
- Un número de 8 a 16 bits en cualquier sistema de numeración que representará dirección o dato según el caso:

n → decimal
$n → hexadecimal
%n → binario
8n → octal

Al final del listado van dos tablas con los nombres de las variables y etiquetas utilizadas y su valor (? indica que son sólo orientativas)

```
:ASM
          *****************************
          *                           *
          *      MONITOR MICRO        *
          *      -------------        *
          *                           *
          *                           *
          *****************************
          *
          *  VARIABLES DEL SISTEMA
          *     INCLUIDA EN PAG. 0
          *
          XREG      EQU  $0001
          YREG      EQU  XREG+1
          STATUS    EQU  YREG+1
          STKP      EQU  STATUS+1
          ACC       EQU  STKP+1
          DIRL      EQU  ACC+1
          DIRH      EQU  DIRL+1
          MACBF     EQU  DIRH+1
          MINBF     EQU  MACBF+1
          OUTBF     EQU  MINBF+1
          AUX       EQU  OUTBF+1
          AUX1      EQU  AUX+1
          USIRQ     EQU  AUX1+1
          CENFLAG   EQU  USIRQ+1
          CENBUFF   EQU  CENFLAG+1
          RSFLAG    EQU  CENBUFF+2
          RSBUFF    EQU  RSFLAG+1
          FUNC      EQU  RSBUFF+1
          IRUCR     EQU  FUNC+2
          HRS       EQU  IRUCR+1
          HRS10     EQU  HRS+1
          MIN       EQU  HRS10+1
          MIN10     EQU  MIN+1
          SECS      EQU  MIN10+1
          SECS10    EQU  SECS+1
          HRSP      EQU  SECS10+1
          MINP      EQU  HRSP+1
          SECSP     EQU  MINP+1
          USREL     EQU  SECSP+1
          INTCNT    EQU  USREL+2
          *
          * DEFINICION DE LOS VALORES
          * DE  LAS TECLAS DE FUNCION
          *
          MAS       EQU  $10
          MENOS     EQU  $11
          PUN       EQU  $12
          DIR       EQU  $13
          DAT       EQU  $14
          *
          * DIRECCION DE LOS REGISTROS
          *         DEL RIOT
          *
          PADR      EQU  $2080       ;DIRECCION PORT A RIOT
          PADDR     EQU  $2081       ;DIRECCION PROGRAMAR PORT A RIOT
          PBDR      EQU  $2082       ;DIRECCION PORT B RIOT
          PBDDR     EQU  $2083       ;DIRECCION PROGRAMAR PORT B RIOT
          PFNSI     EQU  $2084       ;FLANCO NEGATIVO SIN INTER.
          PFPSI     EQU  $2085       ;FLANCO POSITIVO SIN INTER.
          PFNCI     EQU  $2086       ;FLANCO NEGATIVO CON INTER.
          PFPCI     EQU  $2087       ;FLANCO POSITIVO CON INTER.
          T1SI      EQU  $2094       ;REG CONTADOR 1 SIN INTER.
          T8SI      EQU  $2095       ;REG CONTADOR 8 SIN INTER.
          T64SI     EQU  $2096       ;REG CONTADOR 64 SIN INTER.
          T1024SI   EQU  $2097       ;REG CONTADOR 1024 SIN INTER.
          T1CI      EQU  $209C       ;REG CONTADOR 1 CON INTER.
          T8CI      EQU  $209D       ;REG CONTADOR 8 CON INTER.
          T64CI     EQU  $209E       ;REG CONTADOR 64 CON INTER.
          T1024CI   EQU  $209F       ;REG CONTADOR 1024 CON INTER.
          *
          *  DIRECCION DE LOS REGISTROS
          *          DE LA VIA
          *
          RBV       EQU  $2100       ;PORT B DE VIA
          RAV       EQU  $2101       ;PORT A DE VIA CON HANDSHAKE
          DRBV      EQU  $2102       ;PROGRAMACION PORT B DE VIA
          DRAV      EQU  $2103       ;PROGRAMACION PORT A DE VIA
          CT1BV     EQU  $2104       ;PROGRAMACION BAJA T1
          CT1AV     EQU  $2105       ;PROGRAMACION ALTA T1
          RT1B      EQU  $2106       ;REGISTRO T1 BAJO
          RT1A      EQU  $2107       ;REGISTRO T1 ALTO
          RT2B      EQU  $2108       ;REGISTRO T2 BAJA
          RT2A      EQU  $2109       ;REGISTRO T2 ALTA
          SRV       EQU  $210A       ;REGISTRO DESPLAZAMIENTO
          ACRV      EQU  $210B       ;REGISTRO AUX. DE CONTROL
          PCRV      EQU  $210C       ;REGISTRO DE CONTROL
          IFRV      EQU  $210D       ;REGISTRO ALARMA INTER.
          IERV      EQU  $210E       ;REGISTRO ACTIVACION INTER.
          RASV      EQU  $210F       ;PORT A SIN HANDSHAKE
          *
          *    COMIENZO DEL CONTENIDO
          *         DE LA EPROM
          *
                    ORG  $F800
          *
          *  RESERVA DE ESPACIO PARA
          *     LAS RUTINAS DEL
          *         USUARIO
          *
          * LAS RUTINAS DEL SISTEMA NO OCUPAN MAS DE LA
          * MITAD DE LA CAPACIDAD DE LA EPROM POR LO QUE
          * LA PARTE INFERIOR DE LA MISMA ESTA LIBRE
          * PARA LAS APLICACIONES DEL USUARIO
          *
                    DS   $400
          *
          *   TABLA DE LAS DIRECCIONES
          *     DE LAS FUNCIONES DEL
          *          EDITOR
          *
FC86: FC  TABLA     DFB  @MAS1/256
FC87: F1            DFB  @MAS1
FC88: FC            DFB  @MENOS1/256
FC89: EE            DFB  @MENOS1
FC8A: FC            DFB  @PUN1/256
```

```
FC8B: DE                    DFB   #RUNI
FC8C: FD                    DFB   #DIRL/256
FC8D: FD                    DFB   #DATO/256
FC8E: 3A                    DFB   #DIRI
FC8F: 00                    DFB   #DATO
                   *
                   * DEFINICION DE REPRESENTACION
                   *  DE CARACTERES EN EL
                   *       DISPLAY
                   *
FC90: 40 79 24
FC93: 30           CODE     DFB   $40,$79,$24,$30   ;'0','1','2','3'
FC94: 19 12 02
FC97: 78                    DFB   $19,$12,$02,$78   ;'4','5','6','7'
FC98: 00 10 08
FC9B: 03                    DFB   $00,$10,$08,$03   ;'8','9','A','B'
FC9C: 46 24 06
FC9F: 0E                    DFB   $46,$24,$06,$0E   ;'C','D','E','F'
FCA0: 0B 09 00
FCA3: 0C                    DFB   $0B,$09,$00,$0C   ;'A','?','?','P'
FCA4: 12 7F                 DFB   $12,$7F           ;'S',' '
                   ***********************************
                   *                                 *
                   *       RUTINA DE RESET           *
                   *                                 *
                   ***********************************
FCA6: 08           RES      PHP
FCA7: 85 05                 STA   ACC
FCA9: 86 01                 STX   XREG      ; SALVA LOS REGISTROS DEL USUARIO
FCAB: 84 02                 STY   YREG
FCAD: 68                    PLA             ;SACA DIRECCION DEL PROGRAMA
FCAE: 85 03                 STA   STATUS    ;Y EL STATUS DEL STACK
FCB0: 68                    PLA
FCB1: 85 07                 STA   DIRL
FCB3: 68                    PLA
FCB4: 85 08                 STA   DIRH
FCB6: A9 0F                 LDA   #$0F      ;PROGRAMA PORT B RIOT
FCB8: 8D 83 20              STA   PBDDR
FCBB: D8                    CLD             ;CALCULA EN BINARIO
FCBC: 78                    SEI             ;INHIBE INTERRUPCIONES
                   ********************************
                   *                              *
                   *       BUCLE PRINCIPAL        *
                   *                              *
                   ********************************
FCBD: 20 15 FE     ENTRADA  JSR   LDAT
FCC0: 20 D7 FD              JSR   LEEKB     ;LEE TECLA
FCC3: C9 10        ENT1     CMP   #$10      ;SI ES UN NUMERO
FCC5: 30 F6                 BMI   ENTRADA   ;IGNORALO
FCC7: C9 14                 CMP   #$14      ;SI ERROR
FCC9: 10 F2                 BPL   ENTRADA   ;IGNORALO
FCCB: 38                    SEC
FCCC: E9 10                 SBC   #$10      ;TRANSFORMAR TECLA
FCCE: 0A                    ASL             ;EN VALOR
FCCF: AA                    TAX             ;PARA CALCULAR DIRECCION
FCD0: BD 86 FC              LDA   TABLA,X   ;DE LA FUNCION
FCD3: 85 1B                 STA   FUNC
FCD5: E8                    INX
FCD6: BD 86 FC              LDA   TABLA,X
FCD9: 85 1C                 STA   FUNC+1
FCDB: 6C 1B 00              JMP   (FUNC)    ;EJECUTA LA FUNCION
                   ***********************************
                   *                                 *
                   *       RUTINA DE ATENCION        *
                   *                                 *
                   *            A  RUN               *
                   *                                 *
                   ***********************************
FCDE: A6 04        RUNI     LDX   STKP      ; RESTAURA LOS REGISTROS
FCE0: 9A                    TSX
FCE1: A5 08                 LDA   DIRH
FCE3: 48                    PHA
FCE4: A5 07                 LDA   DIRL
FCE6: 48                    PHA
FCE7: A5 03                 LDA   STATUS
FCE9: 48                    PHA
FCEA: A6 01                 LDX   XREG
FCEC: A4 02                 LDY   YREG
FCEE: A5 05                 LDA   ACC
FCF0: 40                    RTI             ; EJECUTA LA RUTINA DEL USUARIO
                   ***********************************
                   *                                 *
                   *       RUTINA DE ATENCION        *
                   *                                 *
                   *            A  +                 *
                   *                                 *
                   ***********************************
FCF1: 18           MAS1     CLC             ;INCREMENTA DIRECCION USUARIO
FCF2: A5 07                 LDA   DIRL
FCF4: 69 01                 ADC   #$01
FCF6: 85 07                 STA   DIRL
FCF8: 90 02                 BCC   MAS2
FCFA: E6 08                 INC   DIRH
FCFC: 4C BD FC     MAS2     JMP   ENTRADA   ;IR BUCLE PRINCIPAL
                   ***********************************
                   *                                 *
                   *       RUTINA DE ATENCION        *
                   *                                 *
                   *            A  -                 *
                   *                                 *
                   ***********************************
FCFF: 38           MENOS1   SEC             ; DECREMENTA DIRECCION USUARIO
FD00: A5 07                 LDA   DIRL
FD02: E9 01                 SBC   #$01
FD04: 85 07                 STA   DIRL
FD06: 90 02                 BCC   MENOS2
FD08: C6 08                 DEC   DIRH
FD0A: 4C BD FC     MENOS2   JMP   ENTRADA   ;VOLVER BUCLE PRINCIPAL
                   ***********************************
                   *                                 *
                   *       RUTINA DE ATENCION        *
                   *                                 *
                   *            A DAT                *
                   ***********************************
FD0D: A0 13        DATO     LDY   #$13      ;PREPARA BUFFER
FD0F: A2 04        DAT1     LDX   #$04
FD11: 86 0B                 STX   MINBF
FD13: A9 06                 LDA   #$06
FD15: 85 0A                 STA   MAXBF
FD17: 20 9A FD              JSR   LEDAT     ;LEE LOS DATOS
FD1A: A2 04                 LDX   #$04
FD1C: 20 30 FD              JSR   AUX1      ;LOS CONVIERTE EN UN BYTE
FD1F: C0 13                 CPY   #$13
FD21: F0 06                 BEQ   DAT2
FD23: 99 01 00              STA   XREG,Y
FD26: 18                    CLC             ;SI ES UN REG. LO ALMACENA
FD27: 90 04                 BCC   DATFIN
FD29: A2 00        DAT2     LDX   #0        ;ALMACENA DATO
FD2B: 81 07                 STA   (DIRL,X)
FD2D: 4C BD FC     DATFIN   JMP   ENTRADA
                   *
                   * RUTINA AUXILIAR DE DATO
                   *
FD30: B5 0C        AUX1     LDA   OUTBF,X   ;CONVIERTE DOS NUMEROS
FD32: 0A                    ASL             ;DE CUATRO BITS EN
FD33: 0A                    ASL             ;UNO DE OCHO.
FD34: 0A                    ASL
FD35: 0A                    ASL
```

```
FD36: E8                        INX
FD37: 15 0C                     ORA   OUTBF,X
FD39: 60                        RTS
                  ********************************
                  *                              *
                  *    RUTINA DE ATENCION        *
                  *                              *
                  *          A DIR               *
                  *                              *
                  ********************************
FD3A: 20 D7 FD    DIR1          JSR   LEEKB         ;LEE TECLA
FD3D: C9 10                     CMP   #$10
FD3F: 30 06                     BMI   DIREC         ;VER SI ES DAT
FD41: C5 14                     CMP   DAT           ;VA A DIRDAT
FD43: D0 F5                     BNE   DIR1          ;PROGRAMA EL BUFFER DE SALIDA
FD45: F0 1B                     BEQ   DIRDAT
FD47: A2 00       DIREC         LDX   #$00
FD49: 86 0B                     STX   MINBF
FD4B: A9 04                     LDA   #$04
FD4D: 85 0A                     STA   MAXBF
FD4F: 20 9A FD                  JSR   LEDAT         ;LEE DIRECCION ALMACENADA
FD52: A9 00                     LDA   #$00
FD54: 20 12 00                  JSR   AUX
FD57: 85 08                     STA   DIRH
FD59: E8                        INX
FD5A: 20 12 00                  JSR   AUX
FD5D: 85 07                     STA   DIRL
FD5F: 4C 8D FC                  JMP   ENTRADA       ;VA AL BUCLE PRINCIPAL
                  ********************************
                  *                              *
                  *    RUTINA DE ATENCION        *
                  *                              *
                  *       A DIR DAT              *
                  *                              *
                  ********************************
FD62: 20 D7 FD    DIRDAT        JSR   LEEKB         ;LEE TECLA
FD65: C9 0A                     CMP   #$0A
FD67: D0 03                     BNE   DDAT1         ;VER SI ES EL ACUMULADOR
FD69: A9 04                     LDA   #$04
FD6B: A8                        TAY
FD6C: A9 7F       DDAT1         LDA   #$7F          ;COLOCA ESPACIOS VACIOS
FD6E: 85 12                     STA   AUX           ;EN EL BUFFER DE SALIDA
FD70: 85 0C                     STA   OUTBF
FD72: 85 0D                     STA   OUTBF+1
FD74: 85 0E                     STA   OUTBF+2
FD76: B9 90 FC                  LDA   CODE,Y
FD79: 85 0F                     STA   OUTBF+3
FD7B: A2 04                     LDX   #$04
FD7D: 20 2C FE                  JSR   GUARDA        ;LO ESCRIBE
FD80: 20 B7 FD                  JSR   ESCRIBE
FD83: 20 D7 FD                  JSR   LEEKB         ;LEE TECLA
FD86: C5 14                     CMP   DAT           ;VE SI ES DAT
FD88: F0 08                     BEQ   DDAT2
FD8A: 48                        PHA
FD8B: 20 15 FE                  JSR   LDAT
FD8E: 68                        PLA                 ;CONVIERTE DATOS PARA DISPLAY
FD8F: 4C C3 FC                  JMP   ENT1
FD92: A4 12       DDAT2         LDY   AUX           ;IR A EJECUTAR DIRDAT
FD94: 20 0F FC                  JSR   DAT1
FD97: 4C 8D FC                  JMP   ENTRADA
                  ********************************
                  *                              *
                  *    LEE UNA DIRECCION         *
                  *                              *
                  *      O UN DATO DEL           *
                  *                              *
                  *         TECLADO              *
                  *                              *
                  ********************************
```



```
FD9A: A6 0B       LEDAT         LDX   MINBF
FD9C: 20 D7 FD    LE1           JSR   LEEKB         ;LEE TECLA
FD9F: C5 11                     CMP   MENOS         ;COMPRUEBA SI ES MENOS
FDA1: F0 09                     BEQ   BK
FDA3: 95 0C                     STA   OUTBF,X
FDA5: 20 B7 FD                  JSR   ESCRIBE
FDA8: E8                        INX
FDA9: E4 0A                     CPX   MAXBF
FDAB: 60                        RTS
FDAC: A9 7F       BK            LDA   #$7F          ;SI ES MENOS
FDAE: 95 0C                     STA   OUTBF,X       ;RETROCEDE UNO MAS
FDB0: CA                        DEX
FDB1: E4 0B                     CPX   MINBF
FDB3: F0 E5                     BEQ   LEDAT
FDB5: D0 E5                     BNE   LE1
                  ********************************
                  *                              *
                  *   RUTINA DE ESCRITURA        *
                  *                              *
                  *     EN EL DISPLAY            *
                  *                              *
                  ********************************
FDB7: A9 7F       ESCRIBE       LDA   #$7F          ;PROGRAMA PORT A
FDB9: 8D 01 20                  STA   PADDR
FDBC: A2 05                     LDX   #$05          ;PREPARA BUFFER
FDBE: B4 0C       ESC1          LDY   OUTBF,X       ;LEE CODIGO
FDC0: B9 90 FC                  LDA   CODE,Y        ;CARGA CODIGO DISPLAY
FDC3: 8D 00 20                  STA   PADR          ;ESCRIBELO
FDC6: A0 7F       UP            LDY   #$7F          ;ESPERA UN RATO
FDC8: 88                        DEY
FDC9: 10 FB                     BPL   UP
FDCB: 8C 00 20                  STY   PADR          ;REPROGRAMA PORT A
FDCE: A9 06                     LDA   #$06
FDD0: 8D 02 20                  STA   PBDR          ;REPROGRAMA PORT B
FDD3: CA                        DEX
FDD4: 10 E8                     BPL   ESC1          ;FIN ?
FDD6: 60                        RTS
                  ********************************
                  *                              *
                  *     LECTURA DE UNA           *
                  *                              *
                  *         TECLA                *
                  *                              *
                  ********************************
FDD7: A9 00       LEEKB         LDA   #$00          ;PROGRAMA PORT
FDD9: 8D 01 20                  STA   PADDR
FDDC: A2 06                     LDX   #$06
FDDE: 8E 02 20                  STX   PBDR
FDE1: AD 00 20    LEE1          LDA   PADR
FDE4: D0 0B                     BNE   SI
FDE6: E8                        INX
FDE7: E0 09                     CPX   #$09
FDE9: D0 F6                     BNE   LEE1
FDEB: 20 B7 FD                  JSR   ESCRIBE
FDEE: 4C D7 FD                  JMP   LEEKB
FDF1: 86 13       SI            STX   AUX1
FDF3: 48                        PHA
FDF4: 20 B7 FD                  JSR   ESCRIBE
FDF7: 68                        PLA
FDF8: A6 13                     LDX   AUX1
FDFA: A0 7F       LEE2          LDY   #$7F
FDFC: 88                        DEY
FDFD: 10 FB                     BPL   LEE2
FDFF: AD 00 20                  LDA   PADR
FE02: 85 12                     STA   AUX
FE04: 8A                        TXA
FE05: 38                        SEC
FE06: E9 06                     SBC   #$06
FE08: F0 08                     BEQ   LEFIN
```

```
FE0A: 18           LEE3    CLC
FE0B: A9 0A                LDA  #$0A
FE0D: 65 12                ADC  AUX
FE0F: CA                   DEX
FE10: D0 F8                BNE  LEE3
FE12: A5 12        LEFIN   LDA  AUX
FE14: 60                   RTS
                   ****************************
                   *                          *
                   *      CARGA DATOS EN      *
                   *                          *
                   *    BUFFER DEL DISPLAY    *
                   *                          *
                   ****************************
FE15: A5 08        LDAT    LDA  DIRH       ;ALMACENA PARTE ALTA Y
FE17: A2 00                LDX  #$00       ;BAJA DE DIRECCION EN EL
FE19: 20 2C FE             JSR  GUARDA     ;BUFFER DE SALIDA
FE1C: A5 07                LDA  DIRL
FE1E: 20 2C FE             JSR  GUARDA
FE21: A0 00                LDY  #0
FE23: B1 07                LDA  (DIRL),Y   ;CARGA CONTENIDO DIRECCION
FE25: 20 2C FE             JSR  GUARDA
FE28: 20 B7 FD             JSR  ESCRIBE
FE2B: 60                   RTS
                   *
                   *    RUTINA AUXILIAR DE LDAT
                   *
FE2C: 48           GUARDA  PHA             ;CONVIERTE NUMERO
FE2D: 29 F0                AND  #$F0       ;DE OCHO BITS
FE2F: 0A                   ASL             ;EN DOS DE CUATRO BITS
FE30: 0A                   ASL
FE31: 0A                   ASL
FE32: 0A                   ASL
FE33: 95 0C                STA  OUTBF,X
FE35: E8                   INX
FE36: 68                   PLA
FE37: 29 0F                AND  #$0F
FE39: 95 0C                STA  OUTBF,X
FE3B: E8                   INX
FE3C: 60                   RTS
                   ********************************
                   *                              *
                   *      PROG. CENTRONICS        *
                   *                              *
                   ********************************
                   *
                   * PROGRAMACION DEL PORT A
                   *    COMO ENTRADA/SALIDA
                   *    INTERFACE CENTRONICS
                   *
                   * PARA LEER EL PUNTO DE ENTRADA
                   *         ES INCENT
                   *
                   *  PARA ESCRIBIR ES OUTCENT
                   *
FE3D: A9 00        INCENT  LDA  #0         ;RUTINA DE ENTRADA
FE3F: 8D 03 21             STA  DRAV       ;PROGRAMA REG. DIRECCION
FE42: 85 15                STA  CENFLAG
FE44: 4C 50 FE             JMP  CONT
FE47: A9 FF        OUTCENT LDA  #$FF       ;RUTINA SALIDA
FE49: 8D 03 21             STA  DRAV       ;PROGRAMA REG. DIRECCION
FE4C: A9 01                LDA  #$01
FE4E: 85 15                STA  CENFLAG
FE50: A9 00        CONT    LDA  #0         ;PARTE COMUN
FE52: 8D 01 21             STA  RAV        ;REGISTRO FLAGS INT.
FE55: AD 0C 21             LDA  PCRV       ;REGISTRO DE CONTROL
FE58: 29 F0                AND  #%11110000
FE5A: 09 08                ORA  #%00001000
FE5C: 8D 0C 21             STA  PCRV
```



```
                   *
FE5F: AD 0B 21             LDA  ACRV       ;REGISTRO AUXILIAR DE CONTROL
FE62: 29 FE                AND  #%11111110
FE64: 09 01                ORA  #%00000001
FE66: 8D 0B 21             STA  ACRV
                   *
FE69: A9 83                LDA  #%10000011
FE6B: 0D 0E 21             ORA  IERV       ;ACTIVACION DE INTER.
FE6E: 8D 0E 21             STA  IERV
FE71: A9 00                LDA  #0
FE73: 8D 0D 21             STA  IFRV
FE76: 58                   CLI
FE77: 60                   RTS
                   ************************************
                   *                                  *
                   *      PROG. RS - 232/V 24         *
                   *                                  *
                   ************************************
                   * HAY DOS PUNTOS DE ENTRADA
                   * INRS PARA LA RUTINA DE ENTRADA
                   * Y OUTRS PARA LA DE SALIDA
                   *
FE78: AD 0B 21     INRS    LDA  ACRV       ;PROGRAMA EL REGISTRO AUXILIAR
FE7B: 29 E1                AND  #%11100001 ;DE CONTROL
FE7D: 09 0E                ORA  #%00001110
FE7F: 8D 0B 21             STA  ACRV
FE82: AD 0C 21             LDA  PCRV       ;IDEM REGISTRO DE CONTROL
FE85: 29 3F                AND  #%00111111
FE87: 09 80                ORA  #%10000000
FE89: 8D 0C 21             STA  PCRV
FE8C: A9 00                LDA  #0         ;RSFLAG=0 -> ENTRADA
FE8E: 85 18                STA  RSFLAG
FE90: 4C AB FE             JMP  CONT1
FE93: AD 0B 21     OUTRS   LDA  ACRV       ;IDEM REGISTRO AUXILIAR DE
FE96: 29 E1                AND  #%11100001 ;CONTROL
FE98: 09 1E                ORA  #%00011110
FE9A: 8D 0B 21             STA  ACRV
FE9D: AD 0C 21             LDA  PCRV       ;IDEM REGISTRO DE CONTROL
FEA0: 29 3F                AND  #%00111111
FEA2: 09 00                ORA  #%00000000
FEA4: 8D 0C 21             STA  PCRV
FEA7: A9 01                LDA  #1         ;RSFLAG=1 -> SALIDA
FEA9: 85 18                STA  RSFLAG
FEAB: A9 00        CONT1   LDA  #0         ;ACTIVAMOS LA INTERRUPCION
FEAD: 8D 0D 21             STA  IFRV       ;CORRESPONDIENTE
FEB0: AD 0E 21             LDA  IERV
FEB3: 09 84                ORA  #%10000100
FEB5: 8D 0E 21             STA  IERV
FEB8: 58                   CLI
FEB9: 60                   RTS
                   ************************************
                   *                                  *
                   *    RUTINA DE ATENCION A          *
                   *                                  *
                   *           IRQ                    *
                   *                                  *
                   ************************************
FEBA: AD 0D 21     IRQQ    LDA  IFRV       ;IRQQ ES LA RUTINA QUE
FEBD: 29 80                AND  #%10000000 ;ATIENDE A LA INTERRUPCION
FEBF: F0 32                BEQ  IRQ1       ;DEL INTERFACE CENTRONICS
FEC1: AD 0D 21             LDA  IFRV
FEC4: 29 02                AND  #%00000010 ;DEPENDIENDO DEL CONTENIDO
FEC6: F0 2B                BEQ  IRQ1       ;DE CENFLAG REALIZA ENTRADA
FEC8: A5 15                LDA  CENFLAG    ;O SALIDA.
FECA: F0 14                BEQ  CENIN
FECC: A5 16        CENOUT  LDA  CENBUFP    ;CENBUFP ES UN VECTOR QUE
FECE: F0 23                BEQ  IRQ1       ;INDICA EL INICIO DEL
FED0: C6 16                DEC  CENBUFP    ;BUFFER DEL INTERFACE
FED2: 8A                   TXA
```

```
FED3: 48                   PHA                ;(DEFINIBLE POR EL USUARIO)
FED4: A2 00                LDX  #0
FED6: A1 16                LDA  (CENBUFP,X)
FED8: 8D 01 21             STA  PAV
FEDB: 68                   PLA
FEDC: 8A                   TXA
FEDD: 4C F3 FE             JMP  IRQ1
FEE0: A5 16       CENIN    LDA  CENBUFP
FEE2: C9 FF                CMP  #$FF
FEE4: F0 0D                BEQ  IRQ1
FEE6: E6 16                INC  CENBUFP
FEE8: 8A                   TXA
FEE9: 48                   PHA
FEEA: A2 00                LDX  #0
FEEC: AD 01 21             LDA  PAV
FEEF: 81 16                STA  (CENBUFP,X)
FEF1: 68                   PLA
FEF2: AA                   TAX
FEF3: AD 0D 21    IRQ1     LDA  IFRV          ;IRQ1 ES LA RUTINA QUE
FEF6: 29 80                AND  #%10000000    ;ATIENDE A LA INTERRUPCION
FEF8: F0 32                BEQ  IRQ2          ;DEL INTERFACE RS - 232
FEFA: AD 0D 21             LDA  IFRV
FEFD: 29 1C                AND  #%00011100    ;RSFLAG INDICA LA DIRECCION
FEFF: F0 2B                BEQ  IRQ2          ;DE TRANSMISION (E/S)
FF01: A5 18                LDA  RSFLAG
FF03: F0 14                BEQ  RSIN          ;LA RUTINA ES SIMILAR A
FF05: A5 19       RSOUT    LDA  RSBUFP        ;LA DEL INTERFACE CENTRONICS
FF07: F0 23                BEQ  IRQ2
FF09: C6 19                DEC  RSBUFP
FF0B: 8A                   TXA
FF0C: 48                   PHA
FF0D: A2 00                LDX  #0
FF0F: A1 19                LDA  (RSBUFP,X)
FF11: 8D 00 21             STA  PBV
FF14: 68                   PLA
FF15: AA                   TAX
FF16: 4C 2C FF             JMP  IRQ2
FF19: A5 19       RSIN     LDA  RSBUFP
FF1B: C9 FF                CMP  #$FF
FF1D: F0 0D                BEQ  IRQ2
FF1F: E6 19                INC  RSBUFP
FF21: 8A                   TXA
FF22: 48                   PHA
FF23: A2 00                LDX  #0
FF25: AD 00 21             LDA  PBV
FF28: 81 19                STA  (RSBUFP,X)
FF2A: 68                   PLA
FF2B: AA                   TAX
FF2C: 6C 1D 00    IRQ2     JMP  (IRUSR)       ;IRUSR ES EL VECTOR DE
                  *  COMIENZO DE RUTINA IRQ DEL USR.
                  **********************************
                  *                                *
                  *    RUTINA DE CONTROL DEL       *
                  *        TEMPORIZADOR            *
                  *                                *
                  **********************************
                  * RUTINA DE APLICACION DE LOS
                  *     TEMPORIZADORES PARA LA
                  *  ACTIVACION DE DISPOSITIVOS
                  *          EXTERIORES
FF2F: A9 FF       INREL    LDA  #IREL/256     ;COLOCA VECTOR USUARIO
FF31: 85 1E                STA  IRUSR+1
FF33: AD BA FF             LDA  IREL
FF36: 85 1D                STA  IRUSR
FF38: AD 0E 21             LDA  IERV          ;INICIALIZA CONTADOR 1 DE VIA
FF3B: 09 C0                ORA  #$C0
FF3D: 8D 0E 21             STA  IERV
FF40: AD 0B 21             LDA  ACRV
FF43: 09 40                ORA  #$40
```



```
FF45: 8D 0B 21             STA  ACRV
FF48: 58                   CLI
FF49: A9 22                LDA  #$22          ;CARGA EL RELOJ CON CUENTA
FF4B: 8D 04 21             STA  CT1BV         ;1/16 DE SEGUNDO
FF4E: A9 F4                LDA  #$F4
FF50: 8D 05 21             STA  CT1AV
FF53: A2 00       LEHORA   LDX  #0            ;INICIALIZA CON LA HORA
FF55: 86 0B                STX  MINBF         ;ACTUAL
FF57: A9 06                LDA  #$06
FF59: 85 0A                STA  MAXBF
FF5B: 20 9A FD             JSR  LEDAT
FF5E: 20 13 00             JSR  AUX1
FF61: 85 1F                STA  HRS
FF63: 20 13 00             JSR  AUX1
FF66: 85 21                STA  MIN
FF68: 20 13 00             JSR  AUX1
FF6B: 85 23                STA  SECS
FF6D: A2 00                LDX  #0            ;LEER LA HORA DE ACTUACION
FF6F: 86 0B                STX  MINBF
FF71: A9 06                LDA  #$06
FF73: 85 0A                STA  MAXBF
FF75: 20 9A FD             JSR  LEDAT
FF78: 20 13 00             JSR  AUX1
FF7B: 85 25                STA  HRSP
FF7D: 20 13 00             JSR  AUX1
FF80: 85 26                STA  MINP
FF82: 20 13 00             JSR  AUX1
FF85: 85 27                STA  SECSP
FF87: 4C A6 FC             JMP  RES           ;REGRESA AL BUCLE PRINCIPAL
                  **********************************
                  *                                *
                  *  RUTINA DE ATENCION INTE-      *
                  *  RRUPCION DE APLICACION        *
                  *     DEL TEMPORIZADOR           *
                  *                                *
                  **********************************
FF8A: 48          IREL     PHA                ;SALVA REGISTROS
FF8B: 8A                   TXA
FF8C: 48                   PHA
FF8D: C6 2A                DEC  INTCNT        ;COMPRUEBA FIN DE CUENTA
FF8F: D0 4D                BNE  FINTR         ;ACTUALIZANDO LOS CONTADORES
FF91: A9 10                LDA  #16           ;DE HORA,MINUTOS Y SEGUNDOS
FF93: 85 2A                STA  INTCNT
FF95: A9 30                LDA  #$30
FF97: E6 23                INC  SECS
FF99: A2 3A                LDX  #$3A
FF9B: E4 23                CPX  SECS
FF9D: D0 3F                BNE  FINTR
FF9F: 85 23                STA  SECS
FFA1: E6 24                INC  SECS10
FFA3: A2 36                LDX  #$36
FFA5: E4 24                CPX  SECS10
FFA7: D0 35                BNE  FINTR
FFA9: 85 24                STA  SECS10
FFAB: E6 21                INC  MIN
FFAD: A2 3A                LDX  #$3A
FFAF: E4 21                CPX  MIN
FFB1: D0 2B                BNE  FINTR
FFB3: 85 21                STA  MIN
FFB5: E6 22                INC  MIN10
FFB7: A2 36                LDX  #$36
FFB9: E4 22                CPX  MIN10
FFBB: D0 21                BNE  FINTR
FFBD: 85 22                STA  MIN10
FFBF: E6 1F                INC  HRS
FFC1: A2 3A                LDX  #$3A
FFC3: E4 1F                CPX  HRS
FFC5: D0 07                BNE  OTRODIA
FFC7: 85 1F                STA  HRS
```

```
FFC9: E6 20      INC  HRS10
FFCB: 4C DE FF   JMP  FINTR
FFCE: A2 34      OTRODIA LDX #$34   ;SI SON 24 HORAS, CAMBIA DIA
FFD0: E4 1F      CPX  HRS
FFD2: D0 0A      BNE  FINTR
FFD4: A2 32      LDX  #$32
FFD6: E4 20      CPX  HRS10
FFD8: D0 04      BNE  FINTR
FFDA: 85 20      STA  HRS10
FFDC: 85 1F      STA  HRS
FFDE: A5 23      FINTR LDA SECS     ;COMPROBAR SI ES LA HORA
FFE0: C5 27      CMP  SECSP         ;PROGRAMADA
FFE2: D0 0F      BNE  F1
FFE4: A5 21      LDA  MIN
FFE6: C5 26      CMP  MINP
FFE8: D0 09      BNE  F1
FFEA: A5 1F      LDA  HRS
FFEC: C5 25      CMP  HRSP
FFEE: D0 03      BNE  F1
FFF0: 6C 28 00   JMP  (USREL)       ;EJECUTA FUNCION USUARIO
FFF3: AD 04 21   F1   LDA CTIBV     ;RESTAURA FLAG INTER.
FFF6: 68         PLA                ;RESTAURA REGISTROS
FFF7: AA         TAX
FFF8: 68         PLA
FFF9: 40         RTI
                 *
                 * DIRECCIONES DE INTERRUPCIONES
                 *
FFFA: A6         NMI   DFB #RES
FFFB: FC               DFB #RES/256
FFFC: A6         RESET DFB #RES
FFFD: FC               DFB #RES/256
FFFE: BA         IRQ   DFB #IRQ0
FFFF: FE               DFB #IRQ0/256

--End assembly--

2048 bytes

Errors: 0

Symbol table - alphabetical order:

  ACC     =$05      ACRV    =$210B    AUX     =$12      AUX1    =$13
  AUXI    =$FD30    BY      =$FDAC    CENBUFF =$16      CENFLAG =$15
  CENIN   =$FEE0  ? CENOUT  =$FECC    CODE    =$FC90    CONT    =$FE50
  CONT1   =$FEAB    CTIAV   =$2105    CTIBV   =$2104    DAT     =$14
  DAT1    =$FD0F    DAT2    =$FD29    DATFIN  =$FD2D    DATO    =$FD0D
  DDAT1   =$FD6C    DDAT2   =$FD92  ? DIR     =$13      DIR1    =$FD3A
  DIRDAT  =$FD62    DIREC   =$FD47    DIRH    =$08      DIRL    =$07
  DRAV    =$2103  ? DRBV    =$2102    ENT1    =$FCC3    ENTRADA =$FCBC
  ESC1    =$FDBE    ESCRIBE =$FDB7    F1      =$FFF3    FINTR   =$FFDE
  FUNC    =$1B      GUARDA  =$FE2C    HRS     =$1F      HRS10   =$20
  HRSP    =$25      IERV    =$210E    IFRV    =$210D  ? INCENT  =$FE30
? INREL   =$FF2F  ? INRS    =$FE78    INTCNT  =$2A      IREL    =$FFBA
? IRQ     =$FFFE    IRQ0    =$FEBA    IRQ1    =$FEF3    IRQ2    =$FF2C
  IPUSR   =$1D      LDAT    =$FE15    LE1     =$FD9C    LEDAT   =$FD9A
  LEE1    =$FDE1    LEE2    =$FDFA    LEE3    =$FE0A    LEEHB   =$FDD7
  LEFIN   =$FE12  ? LEHORA  =$FF53  ? MAS     =$10      MAS1    =$FCF1
  MAS2    =$FCFC    MAXBF   =$0A      MENOS   =$11      MENOS1  =$FCFF
  MENOS2  =$FD0A    MIN     =$21      MIN10   =$22      MINBF   =$0B
  MINP    =$26    ? NMI     =$FFFA    OTRODIA =$FFCE    OUTBF   =$0C
? OUTCENT =$FE47  ? OUTRS   =$FE93    PADDR   =$2081    PADR    =$2080
? PASV    =$210F    PAV     =$2101    PBDDR   =$2083    PBDR    =$2082
  PBV     =$2100    PCRV    =$210C  ? PFNCI   =$2086  ? PFNSI   =$2084
? PFPCI   =$2087  ? PFPSI   =$2085    RES     =$FCA6  ? RESET   =$FFFC
  RSBUFF  =$19      RSFLAG  =$18      RSIN    =$FF19  ? RSOUT   =$FF05
? RT1A    =$2107  ? RT1B    =$2106  ? RT2A    =$2109  ? RT2B    =$2108
? RUN     =$12      RUN1    =$FCDE    SECS    =$23      SECS10  =$24
  SECSP   =$27      SI      =$FDF1  ? SRV     =$210A    STATUS  =$03
  STKP    =$04    ? T1024CI =$209F  ? T1024SI =$2097  ? TICI    =$209C
? TISI    =$2094  ? T64CI   =$209E  ? T64SI   =$2096  ? T8CI    =$209D
? T8SI    =$2095    TABLA   =$FC86    UP      =$FDC4    USIRQ   =$14
  USREL   =$28      XREG    =$01      YREG    =$02

Symbol table - numerical order:

  XREG    =$01      YREG    =$02      STATUS  =$03      STKP    =$04
  ACC     =$05      DIRL    =$07      DIRH    =$08      MAXBF   =$0A
  MINBF   =$0B      OUTBF   =$0C    ? MAS     =$10      MENOS   =$11
  AUX     =$12    ? RUN     =$12      AUX1    =$13    ? DIR     =$13
  USIRQ   =$14      DAT     =$14      CENFLAG =$15      CENBUFF =$16
  RSFLAG  =$18      RSBUFF  =$19      FUNC    =$1B      IPUSR   =$1D
  HRS     =$1F      HRS10   =$20      MIN     =$21      MIN10   =$22
  SECS    =$23      SECS10  =$24      HRSP    =$25      MINP    =$26
  SECSP   =$27      USREL   =$28      INTCNT  =$2A      PADR    =$2080
  PADDR   =$2081    PBDR    =$2082    PBDDR   =$2083  ? PFNSI   =$2084
? PFPSI   =$2085  ? PFNCI   =$2086  ? PFPCI   =$2087  ? TISI    =$2094
? T8SI    =$2095  ? T64SI   =$2096  ? T1024SI =$2097  ? TICI    =$209C
? T8CI    =$209D  ? T64CI   =$209E  ? T1024CI =$209F    PBV     =$2100
  PAV     =$2101  ? DRBV    =$2102    DRAV    =$2103    CTIBV   =$2104
  CTIAV   =$2105  ? RT1B    =$2106  ? RT1A    =$2107  ? RT2B    =$2108
? RT2A    =$2109  ? SRV     =$210A    ACRV    =$210B    PCRV    =$210C
  IFRV    =$210D    IERV    =$210E  ? PASV    =$210F    TABLA   =$FC86
  CODE    =$FC90    RES     =$FCA6    ENTRADA =$FCBC    ENT1    =$FCC3
  RUN1    =$FCDE    MAS1    =$FCF1    MAS2    =$FCFC    MENOS1  =$FCFF
  MENOS2  =$FD0A    DATO    =$FD0D    DAT1    =$FD0F    DAT2    =$FD29
  DATFIN  =$FD2D    AUXI    =$FD30    DIR1    =$FD3A    DIREC   =$FD47
  DIRDAT  =$FD62    DDAT1   =$FD6C    DDAT2   =$FD92    LEDAT   =$FD9A
  LE1     =$FD9C    BY      =$FDAC    ESCRIBE =$FDB7    ESC1    =$FDBE
  UP      =$FDC6    LEEHB   =$FDD7    LEE1    =$FDE1    SI      =$FDF1
  LEE2    =$FDFA    LEE3    =$FE0A    LEFIN   =$FE12    LDAT    =$FE15
  GUARDA  =$FE2C  ? INCENT  =$FE30  ? OUTCENT =$FE47    CONT    =$FE50
? INRS    =$FE78  ? OUTRS   =$FE93    CONT1   =$FEAB    IRQ0    =$FEBA
? CENOUT  =$FECC    CENIN   =$FEE0    IRQ1    =$FEF3  ? RSOUT   =$FF05
  RSIN    =$FF19    IRQ2    =$FF2C  ? INREL   =$FF2F  ? LEHORA  =$FF53
  IREL    =$FFBA    OTRODIA =$FFCE    FINTR   =$FFDE    F1      =$FFF3
? NMI     =$FFFA  ? RESET   =$FFFC  ? IRQ     =$FFFE
```

APENDICE E
LISTADO HEXADECIMAL DEL PROGRAMA MONITOR

Aquí presentamos el listado tal como debe grabarse en la EPROM. Como se observará, no está listado en las mismas direcciones que en el listado fuente; esto se debe a que en el ordenador que utilizamos para desarrollar el programa dichas posiciones están ocupadas por la ROM de dicho ordenador; por ello, se colocó a partir de la dirección $2000. Para grabar la EPROM este detalle es irrelevante.

Desde la dirección $2000 hasta la dirección $2477 no hay programa. El usuario puede colocar aquí sus rutinas antes de grabar la EPROM.

```
2000- FF FF 00 00 FF FF 00 00
2008- FF FF 00 00 FF FF 00 00
2010- FF FF 00 00 FF FF 00 00
2018- FF FF 00 00 FF FF 00 00
2020- FF FF 00 00 FF FF 00 00
2028- FF FF 00 00 FF FF 00 00
2030- FF FF 00 00 FF FF 00 00
2038- FF FF 00 00 FF FF 00 00
2040- FF FF 00 00 FF FF 00 00
2048- FF FF 00 00 FF FF 00 00
2050- FF FF 00 00 FF FF 00 00
2058- FF FF 00 00 FF FF 00 00
2060- FF FF 00 00 FF FF 00 00
2068- FF FF 00 00 FF FF 00 00
2070- FF FF 00 00 FF FF 00 00
2078- FF FF 00 00 FF FF 00 00
2080- FF FF 00 00 FF FF 00 00
2088- FF FF 00 00 FF FF 00 00
2090- FF FF 00 00 FF FF 00 00
2098- FF FF 00 00 FF FF 00 00
20A0- FF FF 00 00 FF FF 00 00
20A8- FF FF 00 00 FF FF 00 00
```

```
20B0- FF FF 00 00 FF FF 00 00
20B8- FF FF 00 00 FF FF 00 00
20C0- FF FF 00 00 FF FF 00 00
20C8- FF FF 00 00 FF FF 00 00
20D0- FF FF 00 00 FF FF 00 00
20D8- FF FF 00 00 FF FF 00 00
20E0- FF FF 00 00 FF FF 00 00
20E8- FF FF 00 00 FF FF 00 00
20F0- FF FF 00 00 FF FF 00 00
20F8- FF FF 00 00 FF FF 00 00
2100- FF FF 00 00 FF FF 00 00
2108- FF FF 00 00 FF FF 00 00
2110- FF FF 00 00 FF FF 00 00
2118- FF FF 00 00 FF FF 00 00
2120- FF FF 00 00 FF FF 00 00
2128- FF FF 00 00 FF FF 00 00
2130- FF FF 00 00 FF FF 00 00
2138- FF FF 00 00 FF FF 00 00
2140- FF FF 00 00 FF FF 00 00
2148- FF FF 00 00 FF FF 00 00
2150- FF FF 00 00 FF FF 00 00
2158- FF FF 00 00 FF FF 00 00
2160- FF FF 00 00 FF FF 00 00
2168- FF FF 00 00 FF FF 00 00
2170- FF FF 00 00 FF FF 00 00
2178- FF FF 00 00 FF FF 00 00
2180- FF FF 00 00 FF FF 00 00
2188- FF FF 00 00 FF FF 00 00
2190- FF FF 00 00 FF FF 00 00
2198- FF FF 00 00 FF FF 00 00
21A0- FF FF 00 00 FF FF 00 00
21A8- FF FF 00 00 FF FF 00 00
21B0- FF FF 00 00 FF FF 00 00
21B8- FF FF 00 00 FF FF 12 37
21C0- FF FF 00 00 FF FF 00 00
21C8- FF FF 00 00 FF FF 00 00
21D0- FF FF 00 00 FF FF 00 00
21D8- FF FF 00 00 FF FF 00 00
21E0- FF FF 00 00 FF FF 00 00
21E8- FF FF 00 00 FF FF 00 00
21F0- FF FF 00 00 FF FF 00 00
21F8- FF FF 00 00 FF FF 87 27
2200- FF FF 00 00 FF FF 00 00
2208- FF FF 00 00 FF FF 00 00
2210- FF FF 00 00 FF FF 00 00
2218- FF FF 00 00 FF FF 00 00
2220- FF FF 00 00 FF FF 00 00
2228- FF FF 00 00 FF FF 00 00
```



```
2230- FF FF 00 00 FF FF 00 00
2238- FF FF 00 00 FF FF 00 00
2240- FF FF 00 00 FF FF 00 00
2248- FF FF 00 00 FF FF 00 00
2250- FF FF 00 00 FF FF 00 00
2258- FF FF 00 00 FF FF 00 00
2260- FF FF 00 00 FF FF 00 00
2268- FF FF 00 00 FF FF 00 00
2270- FF FF 00 00 FF FF 00 00
2278- FF FF 00 00 FF FF 00 00
2280- FF FF 00 00 FF FF 00 00
2288- FF FF 00 00 FF FF 00 00
2290- FF FF 00 00 FF FF 00 00
2298- FF FF 00 00 FF FF 00 00
22A0- FF FF 00 00 FF FF 00 00
22A8- FF FF 00 00 FF FF 00 00
22B0- FF FF 00 00 FF FF 00 00
22B8- FF FF 00 00 FF FF 00 00
22C0- FF FF 00 00 FF FF 00 00
22C8- FF FF 00 00 FF FF 00 00
22D0- FF FF 00 00 FF FF 00 00
22D8- FF FF 00 00 FF FF 00 00
22E0- FF FF 00 00 FF FF 00 00
22E8- FF FF 00 00 FF FF 00 00
22F0- FF FF 00 00 FF FF 00 00
22F8- FF FF 00 00 FF FF 00 00
2300- FF FF 00 00 FF FF 00 00
2308- FF FF 00 00 FF FF 00 00
2310- FF FF 00 00 FF FF 00 00
2318- FF FF 00 00 FF FF 00 00
2320- FF FF 00 00 FF FF 00 00
2328- FF FF 00 00 FF FF 00 00
2330- FF FF 00 00 FF FF 00 00
2338- FF FF 00 00 FF FF 00 00
2340- FF FF 00 00 FF FF 00 00
2348- FF FF 00 00 FF FF 00 00
2350- FF FF 00 00 FF FF 00 00
2358- FF FF 00 00 FF FF 00 00
2360- FF FF 00 00 FF FF 00 00
2368- FF FF 00 00 FF FF 00 00
2370- FF FF 00 00 FF FF 00 00
2378- FF FF 00 00 FF FF 00 00
2380- FF FF 00 00 FF FF 00 00
2388- FF FF 00 00 FF FF 00 00
2390- FF FF 00 00 FF FF 00 00
2398- FF FF 00 00 FF FF 00 00
23A0- FF FF 00 00 FF FF 00 00
23A8- FF FF 00 00 FF FF 00 00
```

```
23B0- FF FF 00 00 FF FF 00 00
23B8- FF FF 00 00 FF FF 7F 12
23C0- FF FF 00 00 FF FF 00 00
23C8- FF FF 00 00 FF FF 00 00
23D0- FF FF 00 00 FF FF 00 00
23D8- FF FF 00 00 FF FF 00 00
23E0- FF FF 00 00 FF FF 00 00
23E8- FF FF 00 00 FF FF 00 00
23F0- FF FF 00 00 FF FF 00 00
23F8  FF FF 00 00 FF FF 57 06
2400- FF FF 00 00 FF FF 00 00
2408  FF FF 00 00 FF FF 00 00
2410- FF FF 00 00 FF FF 00 00
2418- FF FF 00 00 FF FF 00 00
2420- FF FF 00 00 FF FF 00 00
2420- FF FF 00 00 FF FF 00 00
2428  FF FF 00 00 FF FF 00 00
2430- FF FF 00 00 FF FF 00 00
2438- FF FF 00 0C FF FF 00 00
2440- FF FF 00 00 FF FF 00 00
2448  FF FF 00 00 FF FF 00 00
2450- FF FF 00 00 FF FF 00 00
2458  FF FF 00 00 FF FF 00 00
2460- FF FF 00 00 FF FF 00 00
2468- FF FF 00 00 FF FF 00 00
2470- FF FF 00 00 FF FF 00 00
2478- FC E3 FC F1 FC D0 FD FC
2480- 2C FF 40 79 24 30 FC F1
2488- FC FF FC DE FD FD 3A 0D
2490- 40 79 24 30 19 12 02 78
2498- 00 10 08 03 46 24 06 0E
24A0- 08 09 0D 0C 12 7F 08 85
24A8- 05 86 01 84 02 68 85 03
24B0- 68 85 07 68 85 08 A9 0F
24B8- 8D 83 20 D8 78 20 15 FE
24C0- 20 D7 FD C9 10 30 F6 C9
24C8- 14 10 F2 38 E9 10 0A AA
24D0- BD 86 FC 85 1B E8 BD 86
24D8- FC 85 1C 6C 1B 00 A6 04
24E0- BA A5 08 48 A5 07 48 A5
24E8- 03 48 A6 01 A4 02 A5 05
24F0- 40 18 A5 07 69 01 85 07
24F8- 90 02 E6 08 4C BD FC 38
2500- A5 07 E9 01 85 07 90 02
2508- C6 08 4C BD FC A0 13 A2
2510- 04 86 08 A9 06 85 0A 20
2518- 9A FD A2 04 20 30 FD C0
2520- 13 F0 06 99 01 00 18 90
```



```
2528- 04 A2 00 81 07 4C BD FC
2530- 85 0C 0A 0A 0A 0A E8 15
2538- 0C 60 20 D7 FD C9 10 30
2540- 06 C5 14 D0 F5 F0 1B A2
2548- 00 86 0B A9 04 85 0A 20
2550- 9A FD A9 00 20 12 00 85
2558- 08 E8 20 12 00 85 07 4C
2560- BD FC 20 D7 FD C9 0A D0
2568- 03 A9 04 A8 A9 7F 85 12
2570- 85 0C 85 0D 85 0E B9 90
2578- FC 85 0F A2 04 20 2C FE
2580- 20 B7 FD 20 D7 FD C5 14
2588- F0 08 48 20 15 FE 68 4C
2590- C3 FC A4 12 20 0F FD 4C
2598- BD FC A6 0B 20 D7 FD C5
25A0- 11 F0 09 95 0C 20 B7 FD
25A8- E0 E4 0A 60 A9 7F 95 0C
25B0- CA E4 0B F0 E5 D0 E5 A9
25B8- 7F 8D 81 20 A2 05 B4 0C
25C0- B9 90 FC 8D 80 20 A0 7F
25C8- 88 10 FB 8C 80 20 A9 06
25D0- 8D 82 20 CA 10 E8 60 A9
25D8- 00 8D 81 20 A2 06 8E 82
25E0- 20 AD 80 20 D0 0B E8 E0
25E8- 09 D0 F6 20 B7 FD 4C D7
25F0- FD 86 13 48 20 B7 FD 68
25F8- A6 13 A0 7F 88 10 FB AD
2600- 80 20 85 12 8A 38 E9 06
2608- F0 08 10 A9 06 65 12 CA
2610- D0 F8 A5 12 60 A5 08 A2
2618- 00 20 2C FE A5 07 20 2C
2620- FE A0 00 B1 07 20 2C FE
2628- 20 B7 FD 60 48 29 F0 0A
2630- 0A 0A 0A 95 0C E8 68 29
2638- 0F 95 0C E8 60 A9 00 8D
2640- 03 21 85 15 4C 50 FE A9
2648- FF 8D 03 21 A9 01 85 15
2650- A9 00 8D 01 21 AD 0C 21
2658- 29 F0 09 08 8D 0C 21 AD
2660- 0B 21 29 FE 09 01 8D 0B
2668- 21 A9 83 0D 0E 21 8D 0E
2670- 21 A9 00 8D 0D 21 58 60
2678- AD 0B 21 29 E1 09 0E 8D
2680- 0B 21 AD 0C 21 29 0F 09
2688- 80 8D 0C 21 A9 00 85 18
2690- 4C AB FE AD 0B 21 29 E1
2698- 09 1E 8D 0B 21 AD 0C 21
26A0- 29 0F 09 00 8D 0C 21 A9
```

```
26A8- 01 85 18 A9 00 8D 0D 21
26B0- AD 0E 21 09 84 8D 0E 21
26B8- 58 60 AD 0D 21 29 80 F0
26C0- 32 AD 0D 21 29 32 F0 28
26C8- A5 15 F0 14 A5 16 F0 23
26D0- C6 16 8A 48 A2 00 A1 16
26D8- 8D 01 21 68 8A 4C F3 FE
26E0- A5 16 C9 FF F0 0D E6 16
26E8- 8A 48 A2 00 AD 01 21 81
26F0- 16 68 AA AD 0D 21 29 80
26F8- F0 32 AD 0D 21 29 1C F0
2700- 28 A5 18 F0 14 A5 19 F0
2708- 23 C6 19 8A 48 A2 00 A1
2710- 19 8D 00 21 68 AA 4C 2C
2718- FF A5 19 C9 FF F0 0D E6
2720- 19 8A 48 A2 00 AD 00 21
2728- 81 19 68 AA 6C 1D 00 A9
2730- FF 85 1E AD 8A FF 85 1D
2738- AD 0E 21 09 C0 8D 0E 21
2740- AD 0B 21 09 40 8D 0B 21
2748- 58 A9 22 8D 04 21 A9 F4
2750- 8D 05 21 A2 00 86 0B A9
2758- 06 85 0A 20 9A FD 20 13
2760- 00 85 1F 20 13 00 85 21
2768- 20 13 00 85 23 A2 00 86
2770- 0B A9 06 85 0A 20 9A FD
2778- 20 13 00 85 25 20 13 00
2780- 85 26 20 13 00 C5 27 4C
2788- A6 FC 48 8A 48 C6 2A D0
2790- 4D A9 10 C5 2A A9 30 E6
2798- 23 A2 3A E4 20 D0 0F 85
27A0- 23 E6 24 A2 36 E4 24 D0
27A8- 35 85 24 E6 21 A2 3A E4
27B0- 21 D0 28 85 21 E6 22 A2
27B8- 36 E4 22 D0 21 85 22 E6
27C0- 1F A2 3A E4 1F D0 07 85
27C8- 1F E6 20 4C DE FF A2 34
27D0- E4 1F D0 0A A2 32 E4 20
27D8- D0 04 85 20 85 1F A5 23
27E0- C5 27 D0 0F A5 21 C5 26
27E8- D0 09 A5 1F C5 25 D0 03
27F0- 6C 28 00 AD 04 21 60 AA
27F8- 68 40 A6 FC A6 FC BA FE
```


APENDICE F
CODIGOS PROGRAMACION

OPERATION CODE TABLE

[illegible]

[illegible]

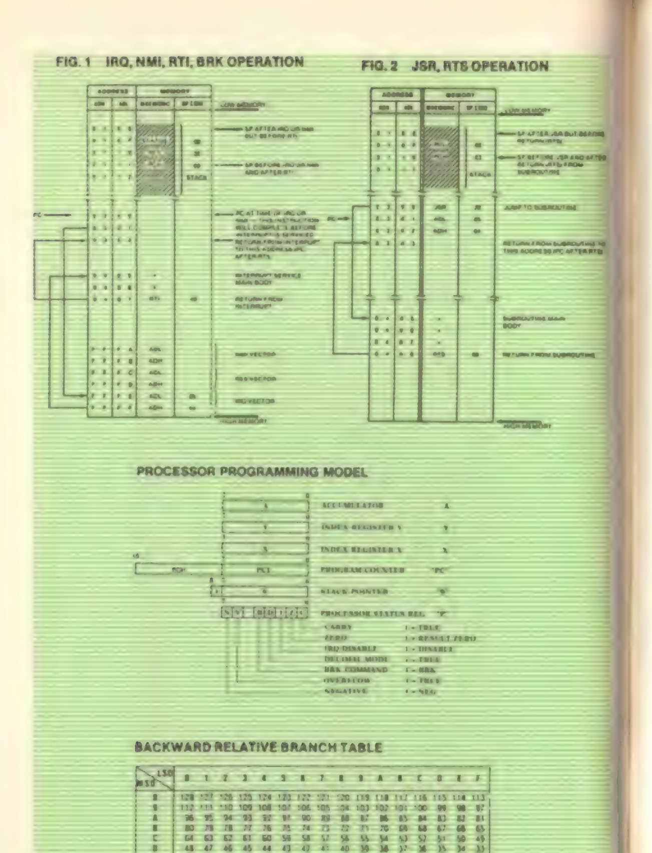
R6500 INSTRUCTION SET

[illegible]	[illegible]

(1) ADD 1 TO N IF PAGE BOUNDARY IS CROSSED.
(2) ADD 1 TO N IF BRANCH OCCURS TO SAME PAGE
ADD 2 TO N IF BRANCH OCCURS TO DIFFERENT PAGE
(3) CARRY NOT = BORROW
(4) IF IN DECIMAL MODE, Z FLAG IS INVALID
ACCUMULATOR MUST BE CHECKED FOR ZERO RESULT

X	INDEX X	+	ADD	M_7	MEMORY BIT 7
Y	INDEX Y	−	SUBTRACT	M_6	MEMORY BIT 6
A	ACCUMULATOR	∧	AND	n	NO. CYCLES
M	MEMORY PER EFFECTIVE ADDRESS	∨	OR	#	NO. BYTES
Ms	MEMORY PER STACK POINTER	∀	EXCLUSIVE OR		

Nota: Continua en la pagina siguiente.



BACKWARD RELATIVE BRANCH TABLE

MSD \ LSD	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8	128	127	126	125	124	123	122	121	120	119	118	117	116	115	114	113
9	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97
A	96	95	94	93	92	91	90	89	88	87	86	85	84	83	82	81
B	80	79	78	77	76	75	74	73	72	71	70	69	68	67	66	65
C	64	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49
D	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33
E	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17
F	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1

FORWARD RELATIVE BRANCH TABLE

MSD \ LSD	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
2	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
3	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
4	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
5	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
6	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
7	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127

COMPARE INSTRUCTION RESULTS

Condition	N	Z	C
A,X, or Y < Memory	1*	0	0
A,X, or Y = Memory	0	1	1
A,X, or Y > Memory	0*	0	1

* N is valid only for 2's complement compare

HEXADECIMAL AND DECIMAL CONVERSION

HEXADECIMAL COLUMNS											
6		5		4		3		2		1	
HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC
0	0	0	0	0	0	0	0	0	0	0	0
1	1,048,576	1	65,536	1	4,096	1	256	1	16	1	1
2	2,097,152	2	131,072	2	8,192	2	512	2	32	2	2
3	3,145,728	3	196,608	3	12,288	3	768	3	48	3	3
4	4,194,304	4	262,144	4	16,384	4	1,024	4	64	4	4
5	5,242,880	5	327,680	5	20,480	5	1,280	5	80	5	5
6	6,291,456	6	393,216	6	24,576	6	1,536	6	96	6	6
7	7,340,032	7	458,752	7	28,672	7	1,792	7	112	7	7
8	8,388,608	8	524,288	8	32,768	8	2,048	8	128	8	8
9	9,437,184	9	589,824	9	36,864	9	2,304	9	144	9	9
A	10,485,760	A	655,360	A	40,960	A	2,560	A	160	A	10
B	11,534,336	B	720,896	B	45,056	B	2,816	B	176	B	11
C	12,582,912	C	786,432	C	49,152	C	3,072	C	192	C	12
D	13,631,488	D	851,968	D	53,248	D	3,328	D	208	D	13
E	14,680,064	E	917,504	E	57,344	E	3,584	E	224	E	14
F	15,728,640	F	983,040	F	61,440	F	3,840	F	240	F	15
7654		3210		7654		3210		7654		3210	
Byte				Byte				Byte			

POWERS OF 2

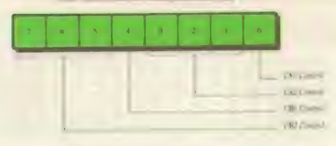
2^n	n
256	8
512	9
1 024	10
2 048	11
4 096	12
8 192	13
16 384	14
32 768	15
65 536	16
131 072	17
262 144	18
524 288	19
1 048 576	20
2 097 152	21
4 194 304	22
8 388 608	23
16 777 216	24

$2^0 = 16^0$
$2^4 = 16^1$
$2^8 = 16^2$
$2^{12} = 16^3$
$2^{16} = 16^4$
$2^{20} = 16^5$
$2^{24} = 16^6$
$2^{28} = 16^7$
$2^{32} = 16^8$
$2^{36} = 16^9$
$2^{40} = 16^{10}$
$2^{44} = 16^{11}$
$2^{48} = 16^{12}$
$2^{52} = 16^{13}$
$2^{56} = 16^{14}$
$2^{60} = 16^{15}$

POWERS OF 16

16^n	n
1	0
16	1
256	2
4 096	3
65 536	4
1 048 576	5
16 777 216	6
268 435 456	7
4 294 967 296	8
68 719 476 736	9
1 099 511 627 776	10
17 592 186 044 416	11
281 474 976 710 656	12
4 503 599 627 370 496	13
72 057 594 037 927 936	14
1 152 921 504 606 846 976	15

6522 PERIPHERAL CONTROL REGISTER (PCR)



CA1 CONTROL

PCR0 = 0 The CA1 Interrupt Flag (IFR1) will be set by a negative transition (high to low) on the CA1 pin.
= 1 The CA1 Interrupt Flag (IFR1) will be set by a positive transition (low to high) on the CA1 pin.

CA2 CONTROL

PCR3	PCR2	PCR1	Mode
0	0	0	CA2 negative edge interrupt (IFR0/ORA clear) mode - Set CA2 interrupt flag (IFR0) on a negative transition of the CA2 input signal. Clear IFR0 on a read or write of the ORA or by writing logic 1 into IFR0.
0	0	1	CA2 negative edge interrupt (IFR0 clear) mode - Set IFR0 on a negative transition of the CA2 input signal. Clear IFR0 by writing logic 1 into IFR0.
0	1	0	CA2 positive edge interrupt (IFR0/ORA clear) mode - Set CA2 interrupt flag (IFR0) on a positive transition of the CA2 input signal. Clear IFR0 on a read or write of the ORA or by writing logic 1 into IFR0.
0	1	1	CA2 positive edge interrupt (IFR0 clear) mode - Set IFR0 on a positive transition of the CA2 input signal. Clear IFR0 by writing logic 1 into IFR0.
1	0	0	CA2 handshake output mode - Set CA2 output low on a read or write of the Peripheral A Output Register. Reset CA2 high with an active transition on CA1.
1	0	1	CA2 pulse output mode - CA2 goes low for one cycle following a read or write of the Peripheral A Output Register.
1	1	0	CA2 low output mode - The CA2 output is held low in this mode.
1	1	1	CA2 high output mode - The CA2 output is held high in this mode.

CB1 CONTROL

PCR4 = 0 The CB1 Interrupt Flag (IFR4) will be set by a negative transition (high to low) on the CB1 pin.
= 1 The CB1 Interrupt Flag (IFR4) will be set by a positive transition (low to high) on the CB1 pin.

CB2 CONTROL

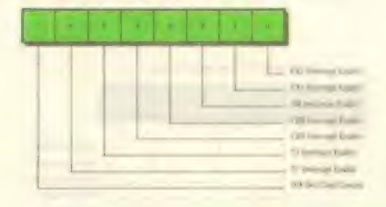
PCR7	PCR6	PCR5	Mode
0	0	0	CB2 negative edge interrupt (IFR3/ORB clear) mode - Set CB2 interrupt flag (IFR3) on a negative transition of the CB2 input signal. Clear IFR3 on a read or write of the ORB or by writing logic 1 into IFR3.
0	0	1	CB2 negative edge interrupt (IFR3 clear) mode - Set IFR3 on a negative transition of the CB2 input signal. Clear IFR3 by writing logic 1 into IFR3.
0	1	0	CB2 positive edge interrupt (IFR3/ORB clear) mode - Set CB2 interrupt flag (IFR3) on a positive transition of the CB2 input a signal. Clear IFR3 on a read or write of the ORB or by writing logic 1 into IFR3.
0	1	1	CB2 positive edge interrupt (IFR3 clear) mode - Set IFR3 on a positive transition of the CB2 input signal. Clear IFR3 by writing logic 1 into IFR3.
1	0	0	CB2 handshake output mode - Set CB2 output low on a write of the Peripheral B Output Register. Reset CB2 high with an active transition on CB1.
1	0	1	CB2 pulse output mode - CB2 goes low for one cycle following a read or write of the Peripheral B Output Register.
1	1	0	CB2 low output mode - The CB2 output is held low in this mode.
1	1	1	CB2 high output mode - The CB2 output is held high in this mode.

6522 INTERRUPT FLAG REGISTER (IFR)



IFR Bit	Set By	Cleared By
0	Active transition on CA2	Reading or writing the ORA ($A001 or $A00F)
1	Active transition on CA1	Reading or writing the ORA ($A001 or $A00F)
2	Completion of eight shifts	Reading or writing the SR ($A00A)
3	Active transition on CB2	Reading or writing the ORB ($A000)
4	Active transition on CB1	Reading or writing the ORB ($A000)
5	Time-out of Timer 2	Reading T2C-L ($A008) or writing T2C-H ($A009)
6	Time-out of Timer 1	Reading T1C-L ($A004) or writing T1L-H ($A005 or $A007)
7	Any IFR bit set with its corresponding IER bit also set	Clearing IFR0-IFR6 ($A00D) or IER0-IER6 ($A00E)

6522 INTERRUPT ENABLE REGISTER (IER)



INTERRUPT ENABLE BITS (IER0-6)

IERn = 0 Disable interrupt
= 1 Enable interrupt

IER SET, CLEAR CONTROL (IER7)

IER7 = 0 For each data bus bit set to logic 1, clear corresponding IER bit.
= 1 For each data bus bit set to logic 1, set corresponding IER bit.

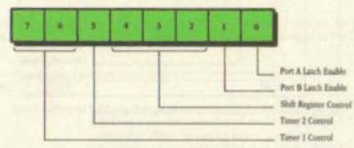
Note: IER7 is active only when R/W = L; when R/W = H, IER7 will read logic 1.

USER R 6522 VERSATILE INTERFACE ADAPTER (VIA)

R6522 M **MEMORY ASSIGNEMENTS**

Location	Function		
A000	Port B Output Data Register (ORB)		
A001	Port A Output Data Register (ORA)		**Controls handshake**
A002	Port B Data Direction Register (DDRB)		0 = Input
A003	Port A Data Direction Register (DDRA)		1 = Output
	Timer	R/W = L	R/W = H
A004	T1	Write T1L-L	Read T1C-L Clear T1 Interrupt Flag
A005	T1	Write T1L-H & T1C-H T1L-L → T1C-L Clear T1 Interrupt Flag	Read T1C-H
A006	T1	Write T1L-L	Read T1L-L
A007	T1	Write T1L-H Clear T1 Interrupt Flag	Read T1L-H
A008	T2	Write T2L-L	Read T2C-L Clear T2 Interrupt Flag
A009	T2	Write T2C-H T2L-L → T2C-L Clear T2 Interrupt Flag	Read T2C-H
A00A	Shift Register (SR)		
A00B	Auxiliary Control Register (ACR)		
A00C	Peripheral Contol Register (PCR)		
A00D	Interrupt Flag Register (IFR)		
A00E	Interrupt Enable Register (IER)		
A00F	Port A Output Data Register (ORA) *No effect on handshake*		

R6522 AUXILIARY CONTROL REGISTER (ACR)



PORT A LATCH ENABLE

ACR0 = 1 Port A latch is enabled to latch input data when CA1 Interrupt Flag (IFR1) is set.
= 0 Port A latch is disabled, reflects current data on PA pins.

PORT B LATCH ENABLE

ACR1 = 1 Port B latch is enabled to latch the voltage on the pins for the input lines or the ORB contents for the output lines when CB1 Interrupt Flag (IFR4) is set.
= 0 Port B latch is disabled, reflects current data on PB pins.

SHIFT REGISTER CONTROL

ACR4	ACR3	ACR2	Mode
0	0	0	Shift Register Disabled.
0	0	1	Shift in under control of Timer 2.
0	1	0	Shift in under control of Ø2.
0	1	1	Shift in under control of external clock.
1	0	0	Free-running output at rate determined by Timer 2.
1	0	1	Shift out under control of Timer 2.
1	1	0	Shift out under control of Ø2.
1	1	1	Shift out under control of external clock.

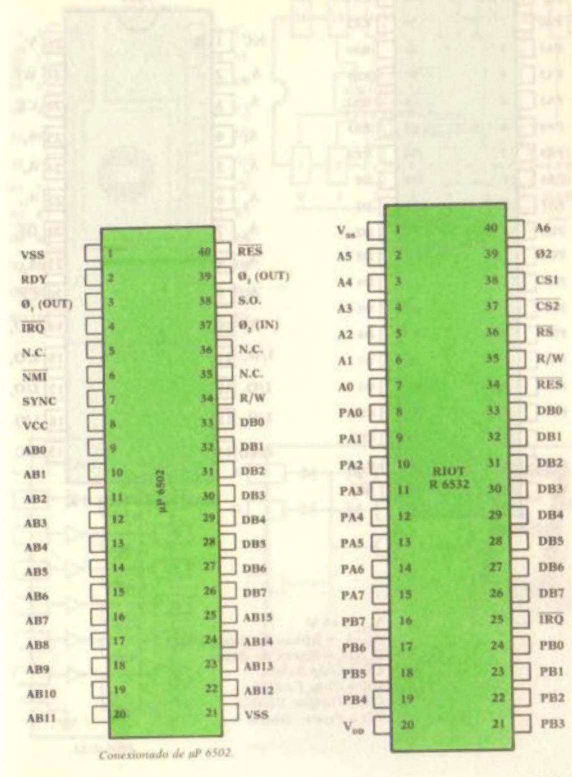
TIMER 2 CONTROL

ACR5 = 0 T2 acts as an interval timer in the one-shot mode.
= 1 T2 counts a predetermined number of pulses on PB6.

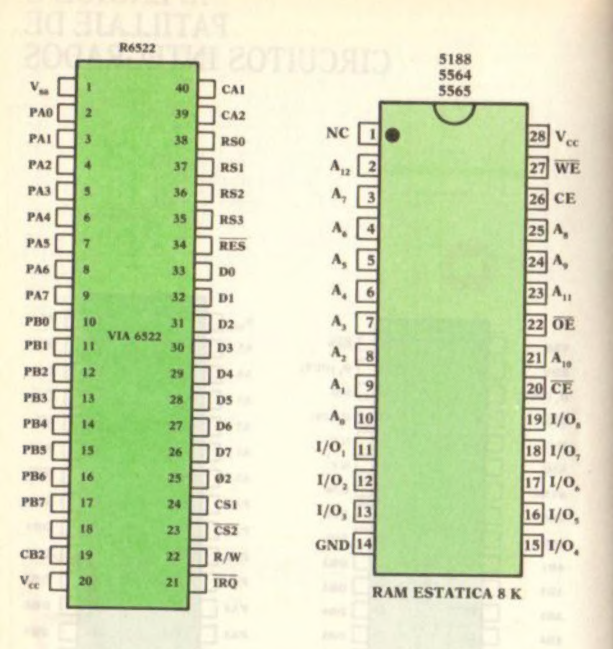
TIMER 1 CONTROL

ACR7	ACR6	Mode
0	0	T1 one-shot mode – Generate a single time-out interrupt each time T1 is loaded. Output to PB7 disabled.
0	1	T1 free-running mode – Generate continuous interrupts. Output to PB7 disabled.
1	0	T1 one-shot mode – Generate a single time-out interrupt and an output pulse on PB7 each time T1 is loaded.
1	1	T1 free-running mode – Generate continuous interrupts and a square wave output on PB7.

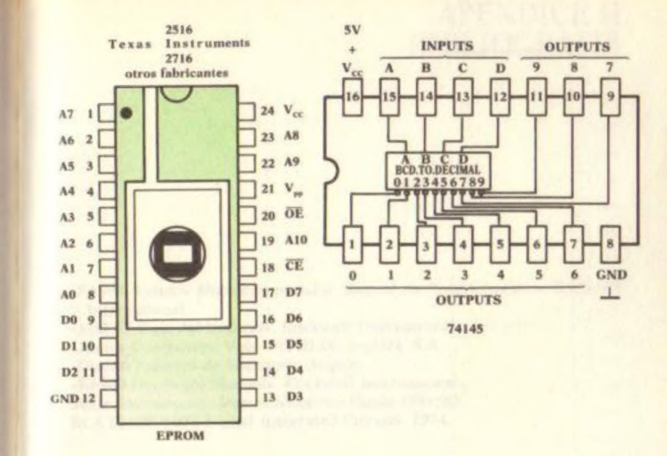
APENDICE G
PATILLAJE DE CIRCUITOS INTEGRADOS

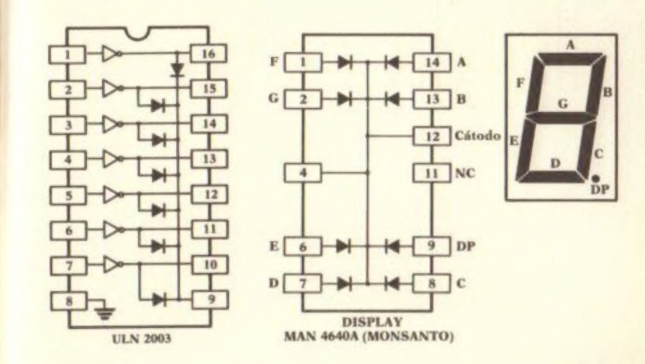


Conexionado de μP 6502.



V_{cc} = +5 V
A_0...A_n = líneas de direcciones
D_0...D_7 = líneas de datos
CS = Chip Select
CE = Chip Enable
OE = Output Enable
PD = Power Down







APENDICE H. BIBLIOGRAFIA

«R6500 Sistema Microcomputador Manual de Programación». Rockwell International.

«AIM 65 Guía del Usuario». Rockwell International.

«Junior Computer». Vols. I-II-III-IV. Ingelek, S.A.

«Diseño Práctico de Sistemas». Angulo.

«R6500 Hardware Manual». Rockwell International.

Texas Instruments, Master Selection Guide 1984/85.

RCA CMOS/MOS Digital Integrated Circuits. 1974.